

Content Manager OnDemand for Multiplatforms
Version 8 Release 5

Indexing Reference



Content Manager OnDemand for Multiplatforms
Version 8 Release 5

Indexing Reference



Note

Before using this information and the product it supports, read the information in “Notices” on page 235.

This edition applies to Version 8 Release 5 of IBM Content Manager OnDemand for Multiplatforms (program number 5724-J33) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC18-9235-02.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

© **Copyright IBM Corporation 1993, 2010.**

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

Contents

About this publication ix

Who should use this publication ix

How this publication is organized ix

ibm.com and related resources x

Support and assistance. xi

Information Center xi

PDF publications. xi

Accessibility information for OnDemand. xi

How to send your comments xi

| What's new in Version 8.5. xi

Part 1. ACIF reference. 1

Chapter 1. Overview 3

About ACIF 4

Indexing concepts 5

Indexing parameters. 5

Converting line data to AFP 7

AFP data 7

Mixed Object Document Content Architecture

Data 7

Line data 8

Mixed-mode data. 8

Unformatted ASCII data 8

AFP resources 8

How OnDemand uses index information 9

ACIF parameters for EBCDIC data 10

Accessing reports 10

Creating indexing parameters 11

Specifying indexing parameters. 11

Determining how literal values are expressed 12

Chapter 2. Using ACIF 15

Example one 15

About the report. 15

Key concepts 17

Defining the application, part 1. 18

Opening the report. 18

Defining triggers 19

Defining fields 19

Defining indexes 20

Displaying triggers, fields, and indexes 21

Indexer Properties 21

Defining the application, part 2. 22

Example two 22

About the report. 22

Key concepts 25

Defining the application, part 1. 26

Opening the report. 27

Defining triggers 27

Defining fields 28

Defining indexes 29

Displaying triggers, fields, and indexes 30

Indexer Properties 30

Defining the application, part 2. 31

Example three 32

About the report. 32

Key concepts 34

Defining the application, part 1. 35

Opening the report. 36

Defining triggers 37

Defining fields 39

Defining indexes 41

Displaying triggers, fields, and indexes 42

Indexer Properties 43

Defining the application, part 2. 43

Example four. 44

About the report. 44

Key concepts 46

Defining the application, part 1. 47

Creating indexing parameters 47

Defining the data format 47

Defining indexing information 47

Defining resource information 48

Defining the application, part 2. 48

Chapter 3. 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 52

Options and values. 52

DCFPAGENAMES 52

Syntax 53

Options and values. 53

EXTENSIONS 53

Syntax 53

Options and values. 53

Related parameters. 55

FDEFLIB 55

Syntax 56

Options and values. 56

Related parameters. 56

FIELD 56

Trigger field syntax. 57

Constant field syntax	58	Options and values.	76
Transaction field syntax	59	Related parameters	76
MASK field syntax	61	MSGDD	76
Related parameters	62	Syntax	77
FILEFORMAT	62	Options and values.	77
Syntax	63	NEWPAGE	77
Options and values.	63	Syntax	77
FONTLIB	63	Options and values.	77
Syntax	64	Related parameters	77
Options and values.	64	OUTEXIT	77
Related parameters	64	Syntax	78
FORMDEF.	64	Options and values.	78
Syntax	65	OUTPUTDD	78
Options and values.	65	Syntax	78
Related parameters	65	Options and values.	78
GROUPMAXPAGES	65	OVLYLIB	78
Syntax	65	Syntax	79
Options and values.	66	Options and values.	79
Related parameters	66	Related parameters	79
GROUPNAME	66	PAGEDEF	79
Syntax	66	Syntax	80
Options and values.	66	Options and values.	80
Related parameters	66	Related parameters	80
IMAGEOUT	66	PARMDD	80
Syntax	67	Syntax	80
Options and values.	67	Options and values.	80
Related parameters	67	PDEFLIB	81
INDEX	67	Syntax	81
Syntax	67	Options and values.	81
Options and values.	68	Related parameters	81
Examples	70	PRMODE	81
Related parameters	70	Syntax	82
INDEXDD.	70	Options and values.	82
Syntax	71	Related parameters	82
Options and values.	71	PSEGLIB	82
INDEXOBJ	71	Syntax	83
Syntax	71	Options and values.	83
Options and values.	71	Related parameters	83
Related parameters	72	RESEXIT	83
INDEXSTARTBY.	72	Syntax	83
Syntax	73	Options and values.	84
Options and values.	73	RESFILE	84
INDEXEXIT.	73	Syntax	84
Syntax	73	Options and values.	84
Options and values.	73	RESLIB.	84
INPEXIT	73	Syntax	85
Syntax	74	Options and values.	85
Options and values.	74	Related parameters	85
INPUTDD.	74	RESOBJDD	85
Syntax	74	Syntax	86
Options and values.	74	Options and values.	86
INSERTIMM	74	RESTYPE	86
Syntax	75	Syntax	86
Options and values.	75	Options and values.	86
Related parameters	75	Related parameters	88
LINECNT	75	TRACE.	88
Syntax	75	Syntax	88
Options and values.	75	Options and values.	88
Related parameters	76	TRC	89
MCF2REF	76	Syntax	89
Syntax	76	Options and values.	89

Related parameters	89
TRIGGER	89
Syntax	90
Options and values	90
Notes	92
Examples	92
Related parameters	93
UNIQUEBNGS	93
Syntax	93
Options and values	93
Related parameters	94
USERLIB	94
Syntax	94
Options and values	94
USERMASK	95
Syntax	95
Options and values	95
Examples	95
Related parameters	96
USERPATH	96
Syntax	96
Options and values	96

Chapter 4. Messages. 97

Chapter 5. User exits and attributes of the input file 99

User programming exits	99
Input record exit	100
Using the user input record exits	103
Index record exit	104
Output record exit.	107
Resource exit	109
Non-Zero return codes	113
Attributes of the input file	113

Chapter 6. Hints and tips 117

Working with control statements that contain numbered lines.	117
Placing TLEs in named groups	117
Working with file transfer	118
About ANSI and machine carriage controls	119
Common methods of transferring files	120
Physical media	120
PC file transfer program.	120
FTP	121
Download	121
Other considerations for transferring files	121
Using the Invoke Medium Map (IMM) structured field	122
Indexing considerations	122
Concatenating resources to an AFP file	123
Specifying the IMAGEOUT parameter	123
Running ACIF with inline resources	124
Writing inline resources to the output file	124

Chapter 7. ACIF data stream information 125

Tag Logical Element (TLE) structured field	125
Format of the resource file	126

Begin Resource Group (BRG) structured field	127
Begin Resource (BR) structured field.	127
End Resource (ER) and End Resource Group (ERG) structured fields	127
Understanding how ACIF processes fully composed AFP files	127

Chapter 8. Format of the ACIF index object file 129

Group-level Index Element (IEL) structured field	129
Page-level Index Element (IEL) structured field	130
Begin Document Index (BDI) structured field.	130
Index Element (IEL) structured field.	130
Tag Logical Element (TLE) structured field	131
End Document Index (EDI) structured field	132

Chapter 9. Format of the ACIF output document file 133

Page groups.	134
Begin Document (BDT) structured field.	135
Begin Named Group (BNG) structured field	135
Tag Logical Element (TLE) structured field	135
Begin Page (BPG) structured field	136
End Named Group (ENG), End Document (EDT), and End Page (EPG) structured fields	136
Output MO:DCA-P data stream	136
Composed Text Control (CTC) structured field	136
Map Coded Font (MCF) Format 1 structured field	136
Map Coded Font (MCF) Format 2 structured field	136
Presentation Text Data Descriptor (PTD) Format 1 structured field	137
Inline resources.	137
Page definitions	137

Chapter 10. Using ACIF in z/OS 139

Sample JCL	139
About the JCL statements	139
ACIF parameters	140
Syntax Rules	141
JCL and ACIF parameters	141
z/OS libraries	143
ACIF output.	143
Concatenating files	143

Part 2. Generic indexer reference 145

Chapter 11. Overview 147

Loading data	147
Processing AFP data	148

Chapter 12. Specifying parameters 151

CODEPAGE:	151
Syntax.	151
Options and values	151
Example	151
COMMENT:.	152
Syntax.	152

Options and values	152
Example	152
GROUP_FIELD_NAME:	152
Syntax.	152
Options and values	152
Example	152
GROUP_FIELD_VALUE:	153
Syntax.	153
Options and values	153
Example	153
GROUP_FILENAME:	153
Syntax.	154
Options and values	154
Example	154
GROUP_LENGTH:	155
Syntax.	155
Options and values	155
Example	155
GROUP_OFFSET:	155
Syntax.	155
Options and values	156
Example	156

Chapter 13. Parameter file examples 157

Part 3. IBM Content Manager OnDemand PDF Indexer for Multiplatforms reference 159

Chapter 14. Overview 161

What is the IBM Content Manager OnDemand PDF Indexer for Multiplatforms?	161
How OnDemand uses index information	163
Indexing input data	164
Indexing concepts	164
Coordinate system.	164
Indexing parameters	164
Indexing with Metadata Indexes	167
How to create indexing parameters	168
Processing PDF input files with the graphical indexer	169
PDF Resource Collection	171

Chapter 15. PDF indexing system requirements 173

Adobe software requirements	173
Specifying the location of Adobe fonts	173
PDF indexing limitations	174
Input data requirements.	175
National language support for indexed PDF documents	175

Chapter 16. Parameter reference . . . 177

BOOKMARKS	177
Syntax.	177
Options and values	177
COORDINATES	177
Syntax.	178
Options and values	178

FIELD.	178
Trigger field syntax	178
Constant field syntax.	180
Related parameters	181
FONTLIB.	181
Syntax.	182
Options and values	182
INDEX	182
Syntax.	182
Options and values	182
Examples.	183
Related parameters	183
INDEXDD	183
Syntax.	184
Options and values	184
INDEXMODE	184
Syntax.	184
Options and values	184
Examples.	184
INDEXSTARTBY	184
Syntax.	185
Options and values	185
INPUTDD	185
Syntax.	186
Options and values	186
MSGDD	186
Syntax.	186
Options and values	186
OUTPUTDD.	186
Syntax.	187
Options and values	187
PARMDD	187
Syntax.	187
Options and values	187
REMOVERES	187
Syntax.	188
Options and values	188
RESOBJDD	188
Syntax.	188
Options and values	188
RESTYPE.	189
Syntax.	189
Options and values	189
TEMPDIR	189
Syntax.	189
Options and values	189
TRACEDD parameter	189
TRIGGER	189
Syntax.	190
Options and values	190
Examples.	191
Related parameters	191

Chapter 17. Message reference. . . . 193

Chapter 18. ARSPDOCI reference. . . 195

Purpose	195
Syntax.	195
Description	195
Parameters	196

Files	196
Chapter 19. ARSPDUMP reference	197
Purpose	197
Syntax.	197
Description	197
Parameters	197
Examples.	198
Files	198
Chapter 20. Trace facility	199
Part 4. Xenos transforms	201
Chapter 21. Understanding Xenos	203
Chapter 22. Xenos transforms	205
Convert data streams.	205
AFP data to PDF	205
Metacode to AFP	205
Metacode to Metacode	205
Metacode to PDF	206
PCL to PDF	206
Index documents	206
Indexing with data values	207
Collect resources	207

Summary.	208
Chapter 23. Loading data	209
Chapter 24. How to specify parameters to the Xenos transforms	211
Processing sample data	211
Specifying parameters to OnDemand	212
Chapter 25. JS program reference	215
Purpose	215
Syntax.	215
Description	215
Parameters	216
Examples.	217
Generating the locations of text strings on a page	218
Indexing and converting input data	222
Loading Metacode documents in large object format.	230
Notices	235
Trademarks	237
Index	239

About this publication

This book contains information about indexing methods, preparing index data, and using tools to index reports that you plan to store in and retrieve from IBM® Content Manager OnDemand for Multiplatforms, Version 8 Release 5 (OnDemand).

Note: The term *Windows® client* refers to the Content Manager OnDemand client program. The term *Windows server* refers to the Content Manager OnDemand server program.

Who should use this publication

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

How this publication is organized

This book is organized in the following parts. Each part contains information about one of the indexing tools supported by Content Manager OnDemand:

- 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). This part contains the following sections:
 - Chapter 1, “Overview,” on page 3 introduces ACIF and describes how you can use ACIF to index data
 - Chapter 2, “Using ACIF,” on page 15 provides examples of using ACIF to index different types of data
 - Chapter 3, “Parameter reference,” on page 49 describes the parameters, options, and values that you can specify to index data, collect AFP resources, and convert line data to AFP data
 - Chapter 4, “Messages,” on page 97 lists the messages that can be generated by ACIF when processing input data
 - Chapter 5, “User exits and attributes of the input file,” on page 99 describes user programming exits, non-zero return codes, and the attributes of the input print file
 - Chapter 6, “Hints and tips,” on page 117 provides information that might be helpful when using ACIF
 - Chapter 7, “ACIF data stream information,” on page 125 contains information about Tag Logical Element (TLE) structure fields, formats of the resource file, and how ACIF processes fully-composed AFP files
 - Chapter 8, “Format of the ACIF index object file,” on page 129 describes the format of the ACIF index object file
 - Chapter 9, “Format of the ACIF output document file,” on page 133 describes the format of the ACIF output document file
 - Chapter 10, “Using ACIF in z/OS,” on page 139 provides information about using ACIF in the z/OS® environment
- Part 2, “Generic indexer reference,” on page 145 describes how to use the Content Manager OnDemand Generic indexer to specify index data for other types of input files (input files that do not contain AFP data, line data, PDF data, or Metacode/DJDE data). This part contains the following sections:

- Chapter 11, “Overview,” on page 147 introduces the Generic indexer and describes how you can use it to index data
- Chapter 12, “Specifying parameters,” on page 151 describes the parameter file that is used by the Generic indexer
- Chapter 13, “Parameter file examples,” on page 157 shows how to specify indexing information
- Part 3, “IBM Content Manager OnDemand PDF Indexer for Multiplatforms reference,” on page 159 describes how to use the Content Manager OnDemand IBM Content Manager OnDemand PDF Indexer for Multiplatforms to generate index data for Adobe® PDF files. This part contains the following sections:

Chapter 14, “Overview,” on page 161 introduces the IBM Content Manager OnDemand PDF Indexer for Multiplatforms and describes how you can use it to extract index data and generate index data about Adobe PDF input files

Chapter 15, “PDF indexing system requirements,” on page 173 contains information about the system requirements, location of Adobe fonts, system limitations, input data requirements, and NLS considerations

Chapter 16, “Parameter reference,” on page 177 describes the parameters to use with the ARSLOAD program to process your input files

Chapter 17, “Message reference,” on page 193 provides an overview of the messages and error conditions that can be generated by the PDF indexer

Chapter 18, “ARSPDOCI reference,” on page 195 contains information about the ARSPDOCI program

Chapter 19, “ARSPDUMP reference,” on page 197 contains information about the ARSPDUMP program

Chapter 20, “Trace facility,” on page 199 describes the tracing capability of the IBM Content Manager OnDemand PDF Indexer for Multiplatforms
 - Part 4, “Xenos transforms,” on page 201 describes how to use the Xenos transforms to process AFP and Metacode/DJDE data. **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 contains the following sections:

Chapter 21, “Understanding Xenos,” on page 203 provides an overview of the Xenos transforms, explains the functions that the Xenos transforms can perform, and describes different scenarios for processing your input files and documents

Chapter 22, “Xenos transforms,” on page 205 discusses how to use the Xenos transforms to convert data streams, index documents, and collect resources

Chapter 23, “Loading data,” on page 209 explains how the ARSLOAD program processes the files that were created by the Xenos transform

Chapter 24, “How to specify parameters to the Xenos transforms,” on page 211 describes how to specify parameters to the Xenos transforms to assist you with developing indexing parameters and how to specify the parameters that are used by the ARSLOAD program and the Xenos transforms to index and convert input files and load them into the system

Chapter 25, “JS program reference,” on page 215 provides information about the JS program, the main Xenos transform program

ibm.com and related resources

Product support and documentation are available from ibm.com®.

Support and assistance

Product support is available on the Web. Click Support from the product Web site at:

Content Manager OnDemand for Multiplatforms

www.ibm.com/software/data/ondemand/mp/support.html

Information Center

You can view the product documentation in an Eclipse-based information center that you can install when you install the product. By default, the information center runs in a Web server mode that other Web browsers can access. You can also run it locally on your workstation. See the information center at: www.ibm.com/software/data/ondemand/mp/support.html

PDF publications

You can view the PDF files online using the Adobe Acrobat Reader for your operating system. If you do not have Acrobat Reader installed, you can download it from the Adobe Web site at www.adobe.com.

You can find PDF publications for IBM Content Manager OnDemand for Multiplatforms at: <http://www.ibm.com/support/docview.wss?rs=129&uid=swg27012713>

Accessibility information for OnDemand

For complete information about accessibility features that are supported by this product, see your Content Manager OnDemand *Administration Guide*.

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. Visit the IBM Data Management Online Reader's Comment Form (RCF) page at www.ibm.com/software/data/rcf.

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).

If you would like to help IBM make IBM Content Manager OnDemand for Multiplatforms easier to use, take the consumability survey at <http://www.ibm.com/software/data/info/consumability-survey/>.

What's new in Version 8.5

Lightweight Directory Access Protocol (LDAP) Secure Sockets Layer (SSL) support

You can now use LDAP with SSL.

Report Distribution enhancements

IBM Content Manager OnDemand Report Distribution for Multiplatforms has been enhanced to allow support for additional data type conversion engines, that is, from line data to PDF.

| **Enhanced reporting and analysis for managing the Content Manager OnDemand system**

| Content Manager OnDemand now provides ability to analyze system log data and generate daily activity reports.

| **Improved capability and productivity for indexing PDF data into Content Manager OnDemand**

| You can now index PDF documents using metadata values contained within the PDF document.

| **Expanded security capabilities to better comply with Federal Information Processing Standards (FIPS 140-2)**

| Content Manager OnDemand has added additional security to better comply with FIPS 140-2.

| **Additional language support**

| The Content Manager OnDemand server and ODWEK installation programs are translated into 9 additional languages. The Content Manager OnDemand client installation program is translated into 22 additional languages. The Content Manager OnDemand Configurator is translated into 9 additional languages.

| **HP-UX Itanium support for DB2 and Oracle databases**

| Content Manager OnDemand now supports HP-UX Itanium for DB2 and Oracle databases.

| **ARSXML system administration performance improvement**

| Performance improvements have been made to the ARSXML system administration command.

| **Install Anywhere replacing ISMP for servers**

| InstallAnywhere 2009 SP1 has replaced InstallShield Multiplatform (ISMP) as the installer engine for Content Manager OnDemand server and ODWEK on Windows and Unix platforms.

Part 1. ACIF reference

This part provides information about ACIF. You can use ACIF to extract index data from and generate index data about AFP and line data reports. You can also use ACIF to convert line data reports to AFP documents and collect the resources required to view and reprint AFP documents.

Chapter 1. Overview

ACIF is a powerful tool for indexing the print data streams of z/OS application programs, unformatted ASCII data, and ASCII data containing printer control characters that is generated on UNIX[®] or Windows workstations. 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 always 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 line data to AFP.
3. Determine whether you need to generate *page-level* index information. There are two types of page-level information, and different ways to generate the information.

Page-level information in the index file. ACIF can generate this type of page-level information whether or not the input data is being converted to AFP. This type of page-level information is essential for loading OnDemand large objects. This type of page-level information is generated by specifying the INDEXOBJ=ALL parameter.

Page-level information in the output file. This type of page-level information is used in the client to move to specific pages in a document. ACIF can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters, and by creating an index field with the TYPE=PAGE option. For more information, see the discussion of TYPE=PAGE in “INDEX” on page 67.

Note that page-level index information is not stored in the database, and therefore cannot be used to search for and retrieve documents.

4. Examine the input data to determine the resource requirements. Determine the fonts and form and page definitions needed to view and print the data.
5. Create parameters for indexing.
6. Create parameters for converting line data input files to AFP.
7. Create parameters for collecting resources for viewing and printing AFP data.

You can run ACIF on an Content Manager OnDemand library or object server or on a zSeries® system on which the ACIF programs are installed.

- When you run ACIF on an Content Manager OnDemand server, you can invoke it from the command prompt (by using the ARSACIF program) or from the Content Manager OnDemand data loading program (the ARSLOAD program). The information provided in this book assumes that you will use the ARSLOAD program to process input data with ACIF. The ARSLOAD program retrieves the indexing parameters that are used to process the input data from the Content Manager OnDemand application.
- To run ACIF on a zSeries system requires:
 - z/OS Version 2 Release 10 or later or z/OS Version 1 Release 1 or later
 - Print Services Facility™ (PSF) for OS/390® Version 3 Release 3 or later
 - OnDemand version of ACIF, which can be ordered without charge by customers who are entitled to OnDemand. See the README file provided with the OnDemand product package for ordering information.

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 and unformatted ASCII. The converted output can be printed, viewed, and archived on any system supported by OnDemand.
- Collection of AFP resources.

ACIF can determine the resources necessary to print, view, and archive the print data stream and collect the resources from PSF and user libraries. Resources allow 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 (see how ACIF locates the specific input records, later in this section). 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, IBM recommends that you 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 document (for example, a statement). After retrieving a document that contains page-level indexes, the user can move to a specific page by using the page-level indexes. **Note:** ACIF can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters, and by creating an index field with the TYPE=PAGE option. For more information, see the discussion of TYPE=PAGE in “INDEX” on page 67.
- 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.

```

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

```

Figure 1. Indexing a report

The following indexing parameters could be used to generate index data for the report 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
- Unformatted ASCII

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*.

Unformatted ASCII data

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

AFP resources

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

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

Table 1. Collecting Resources

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

The resources that ACIF collects is based on the value of the RESTYPE parameter. When ACIF processes a file, it:

- Identifies the resources requested by the print file.

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

- Creates a resource file.

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

- Calls the specified resource exit for each resource it retrieves.

You can specify the name of a resource exit on the RESEXIT parameter so that ACIF filters out any resources that you do not want included in the resource file.

- Includes the name of the output document in the resource file and the name of the resource file in the output document. This provides a method of correlating resources files with the appropriate output document.

How OnDemand uses index information

Every item stored in Content Manager 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 Content Manager 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.

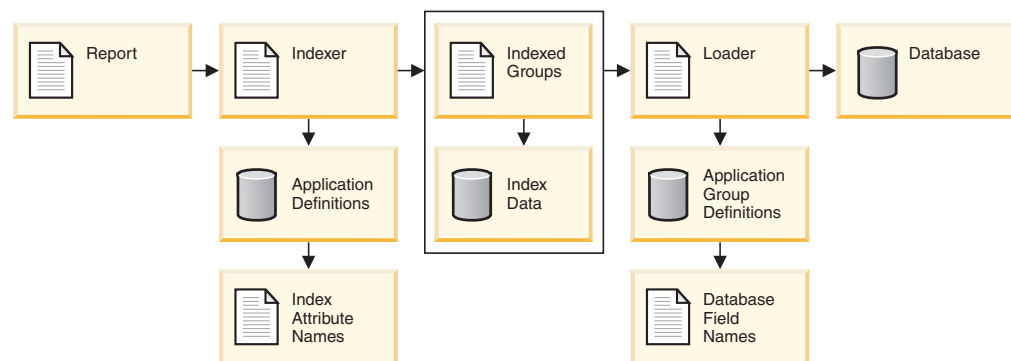


Figure 2. Indexing reports

You typically create an Content Manager OnDemand application for each report that you plan to store in Content Manager 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. Content Manager 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 documents that contain index values (ACIF attribute values) that match the search values entered by the user. OnDemand lists the documents that match the query. When the user selects a document for viewing, the OnDemand client program retrieves the document from cache storage or archive storage. If the document contains page-level indexes that were generated when the report was processed by ACIF, the user can move to a specific page of the document by using the page-level index information.

Note: Only group-level indexes are stored in the database. Page-level indexes are not stored in the database. This means that users cannot use page-level indexes to search for reports that are stored in the system. 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 can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters, and by creating an index field with the TYPE=PAGE option. For more information, see the discussion of TYPE=PAGE in “INDEX” on page 67.

ACIF parameters for EBCDIC data

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

Accessing reports

Reports generated on an z/OS system are typically transmitted to an Content Manager OnDemand server using Download for z/OS (Download), a feature of PSF for z/OS. The report must be transmitted to the server as a binary file to retain the data as EBCDIC. The input data contains variable length records. You must specify FILEFORMAT=RECORD to correctly process the file.

Do not transmit the report to the server as a text file: If you transmit the report as a text file, the data is converted from EBCDIC to ASCII, and carriage controls are inserted into the data. This can affect the ability of OnDemand to index and subsequently read the file.

Creating indexing parameters

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

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

```
index6=X'C39AA2A396948599D5819485',field6 /* CustomerName */
```

Figure 3. Indexing EBCDIC data

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 hexadecimal X'F1' in the first byte, ACIF uses the second trigger to look for the hexadecimal string X'D7C1C7C540F0F0F1' (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 hexadecimal X'F1' in the first byte is considered the anchor record. ACIF uses the anchor record to locate index values. The trigger specifications are expressed in hexadecimal, as follows:

```
trigger1=*,1,X'F1' /* Skip to Channel 1 */  
trigger2=0,72,X'D7C1C7C540F0F0F1' /* PAGE 0001 */
```

Figure 4. 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 hexadecimal string X'839AA2A3958194' (custnam) as the indexing attribute. The index field name is the same as the application group database field name. Locate customer name index values in the second record following the anchor record, starting at byte 40 and extending for 20 bytes. The FIELD and INDEX specifications are expressed as follows:

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

Figure 5. Indexing EBCDIC data (part 2 of 4)

To create the indexing tag for the statement date attribute, define the hexadecimal string X'A28481A385' (sdate) as the indexing attribute. The index field name is the same as the application group database field name. Locate statement date index values in the sixth record following the anchor record, starting at byte 56 and

extending for 10 bytes. The FIELD and INDEX specifications are expressed as follows:

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

Figure 6. Indexing EBCDIC data (part 3 of 4)

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

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

Figure 7. 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.

Determining how literal values are expressed

The way literal values in the input file are defined in ACIF parameters depends on whether the input file contains ASCII or EBCDIC data. If the input file is in ASCII for UNIX or Windows or in EBCDIC for z/OS, then the literal values in the FIELD, INDEX, and TRIGGER parameters must be expressed in character data strings. For example, Figure 8 on page 13 shows part of a parameter file for ASCII input data. The CCTYPE parameter value matches the type of data in the input file, in this case ASCII. The CPGID parameter indicates a code page for the type of data in the input file. The FIELD, INDEX, and TRIGGER parameters are expressed in character data strings because the input file is ASCII and the operating system is UNIX or Windows.

```

/* Example phone bill */
/* DATA CHARACTERISTICS */
CC=yes
CCTYPE=z
CHARS=42B2
CPGID=850
/* FIELD AND INDEX DEFINITION */
FIELD1=13,66,15
FIELD2=0,50,30
FIELD3=1,50,30
FIELD4=2,50,30
FIELD5='1'
INDEX1='Account',FIELD1
INDEX2='Name',FIELD2
INDEX3='Address',FIELD3
INDEX4='City',FIELD4
INDEX5='Date',FIELD5
/* EXIT AND TRIGGER INFORMATION */
TRIGGER1=*,1,'1'
TRIGGER2=13,50,'ACCOUNT'
/* Carriage control used */
/* ASCII ANSI carriage controls */
/* Coded font */
/* Code page identifier */
/* Account data field */
/* Name data field */
/* Address data field */
/* City data field */
/* Date data field */
/* 1st index attribute */
/* 2nd index attribute */
/* 3rd index attribute */
/* 4th index attribute */
/* 5th index attribute */
/* 1st trigger */
/* 2nd trigger */

```

Figure 8. Example of a UNIX or Windows Parameter File for ASCII Input Data

If the input data file is *not* ASCII in UNIX or Windows or *not* EBCDIC in z/OS, then the literal values in the FIELD, INDEX, and TRIGGER parameters must be expressed in hexadecimal strings. For example, Figure 9 shows part of a UNIX parameter file for EBCDIC input data. The CCTYPE parameter value matches the type of data in the input file, in this case EBCDIC. The CPGID parameter indicates a code page for the type of data in the input file. The FIELD, INDEX, and TRIGGER parameters are expressed in hexadecimal strings because the input file is EBCDIC and the operating system is UNIX or Windows.

```

/* Example phone bill */
/* DATA CHARACTERISTICS */
CC=yes
CCTYPE=a
CHARS=GT15
CPGID=037
/* FIELD AND INDEX DEFINITION */
FIELD1=13,66,15
FIELD2=0,50,30
FIELD3=1,50,30
FIELD4=2,50,30
FIELD5=X'F1'
INDEX1=X'C1838396A495A3',FIELD1
INDEX2=X'D5819485',FIELD2
INDEX3=X'C184849985A2A2',FIELD3
INDEX4=X'C389A3A8',FIELD4
INDEX5=X'C481A385',FIELD5
/* EXIT AND TRIGGER INFORMATION */
TRIGGER1=*,1,X'F1'
TRIGGER2=13,50,X'C1C3C3D6E4D5E3'
/* Carriage control used */
/* EBCDIC ANSI carriage controls */
/* Coded font */
/* Code page identifier */
/* Account data field */
/* Name data field */
/* Address data field */
/* City data field */
/* Date data field */
/* 1st index attr (Account) */
/* 2nd index attr (Name) */
/* 3rd index attr (Address) */
/* 4th index attr (City) */
/* 5th index attr (Date) */
/* 1st trigger (1) */
/* 2nd trigger (ACCOUNT) */

```

Figure 9. Example of a UNIX or Windows Parameter File for EBCDIC Input Data

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 10 on page 16 shows a sample page of the loan report, as it appears when viewed using an OnDemand client program.

For faster loading and retrieval, the report should be segmented into 100 page groups when it is loaded into the system. One row should be created 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 11 on page 17 shows the indexer parameters required for ACIF to process the loan report.

Accessing report data

This example shows how to use the OnDemand graphical indexer to process a sample report and create indexing information. The graphical indexer is part of the OnDemand administrative client, a program that runs on a Windows workstation. To process a sample report, you typically create or extract a subset of a complete report. The report in this example was generated on a z/OS system and transferred to the PC as a binary file, and then loaded into the graphical indexer.

The sample data used to create the indexing information must match the actual data to be indexed and loaded into the database. When you load a report into the system, OnDemand uses the indexing parameters, options, and data values that are stored with 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, 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	PENNANT NATIONAL BANK	DATE	10/01/94
BANK	001		TIME	16:03:46
FROM	10/01/94		MODE	9
TO	10/01/94	LOAN DELINQUENCY REPORT	PAGE	00001

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

Figure 10. Loan Report

```

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

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

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

/* RESOURCE INFORMATION */
RESTYPE=NONE /* do not collect 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.

Transaction Field. A field that is used to index a report and that contains one or more columns of sorted data. Because it is not always practical to store every index value in the database, ACIF extracts the first and last sorted values in each group. Depending on the format (ASCII or EBCDIC) of the report data, the data is sorted according to the collating sequence of the code page.

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 OnDemand application identifies the type of data that is stored in the system, the method used to index the data, and other information used to load and view reports. This section provides information about the application for the sample loan report.

General

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

Assign the application to the application group in which the loan report data will be maintained. The application group contains database fields for the report date, beginning loan number, and ending loan number.

View Information

The View Information page is where you specify information needed by OnDemand client programs to display the loan report. Some of the information is also used for the indexing parameters.

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

- Code Page. Set the code page to 500. This is the code page of the data as it is stored in the system and viewed by 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 you specify information that is used to generate index data for the report.

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

- CONVERT=N0. The default value for a line data input file. When loading line data into the system, ACIF does not have to convert the data.
- CPGID=500. The code page of the data as it is stored in the system.
- 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 that contain default values for processing line data.

Next, define additional ACIF parameters, including those that determine the index data that will be extracted from the report and loaded into the database. To do so, you can process sample report data with the OnDemand graphical indexer.

Opening the report

In the Add an Application window, click 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. The

client opens the Indexer Properties dialog box. After you click OK or Cancel, the client 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 that are stored with 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, 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.

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, define a trigger that instructs ACIF to examine the first byte of every input record for the presence of an EBCDIC skip-to-channel one carriage control character (X'F1'). This is the only trigger needed.

Because the graphical indexer displays the report as it is viewed in OnDemand, you 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, ACIF should search every record. The Columns to Search area determines the columns of the trigger record ACIF searches. Set the Columns to Search to Carriage Control so that ACIF searches column one of each record. When you do that, the graphical indexer displays the trigger value (X'F1') that ACIF searches for.

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

Defining fields

ACIF uses fields to determine where to locate index values. For the sample loan report, 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, 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 and select Field to display 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 selected field value.

To save the field information, click OK to add the field and return to the report window.

The second field to define contains the loan number. To support the way that OnDemand should segment and load the data, and the way users will search for reports, define a *transaction field*. A transaction field allows OnDemand to index a group of pages by 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. Select 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 and select Transaction Field 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 the field data is organized. The Mask determines the pattern of symbols that ACIF matches to data located in the field. The number symbol (#) matches any numeric character. 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.

To save the field information, click OK to add the field and return to the report window.

Defining indexes

The next task in defining the indexing parameters for the loan report is to define the indexes. The indexes determine the values that are stored in the database and the type of index. For the loan report, 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. Accept the suggested default, the application group database field name, report_date. When data is loaded 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 the system, you must define at least one group index.) 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 about that later.) The Fields area lists the field parameters that have been defined (in the Fields list) for the report. 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>> to move 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. Because loan number values are not mapped directly to a database field, you must enter your own index name in the Attribute field. Later, on the Load Information page, you will map the index to application group database fields. Type Loan Number in the Attribute field. Because ACIF should extract the beginning and ending loan numbers for a group of pages, select an index Type of GroupRange. Because a grouprange trigger can never break a group, Break must always be set to No. Finally, 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.

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, click the Display and Add Parameters icon to return 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 to highlight the area of the report where the field was defined. Click Field1 again to highlight the area on the next page. Click Field2 to highlight the field in the report. Click Index1 and then click Properties to display the Update an Index dialog box. Click Cancel. Click Trigger1 and then click Properties to display the Update a Trigger dialog box. Click Cancel. Close the Select dialog box.

Indexer Properties

After defining the View Information and the triggers, fields, and indexes, 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 about the format of the input data, the resources required to index the data, the index output file, and the optional user-written programs that may be used to process input, output, and index records and resources. Some of the parameter values are based on the choices that you make on the View Information page. Click Help on each page to display information about the fields. You can also see Chapter 3, "Parameter reference," on page 49 for details.

Click the Output Information tab. Type 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.

Click OK to save the changes and return to the report window. Close the report window. When prompted, click Yes to save your changes and 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 that were added based on the settings in the Indexer Properties dialog box.

Load Information

The Load Information page is where you map the index attribute name that was defined to hold loan number attribute values to application group data base fields. For the loan report, ACIF generates indexes for the first and last loan numbers in a group of pages. The attribute name is Loan Number. OnDemand should 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

Click OK to add the application, update the database, 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 users should view the statements in the same format as the customer's printed copy, ACIF converts the input line data to AFP and collects the resources required to view the statements. Figure 12 on page 23 shows an example of a statement viewed with one of the OnDemand client programs. Figure 13 on page 24 shows what the input data looks like viewed with an ISPF browser on the z/OS system. Because the input data is encoded in EBCDIC, the ACIF trigger and index parameter values must be coded in hexadecimal.

For ease of retrieval, the report should be segmented into groups of pages, with one statement in each group. One index row should be generated for each group. The row contains three user-defined indexes: the account number, the customer's name, and the bill date. Figure 14 on page 25 shows the ACIF indexer parameters required to process the data.

Accessing the report


This example provides instructions about using the OnDemand graphical indexer to process a sample report and create indexing information. The graphical indexer is part of the OnDemand administrative client, a program that runs on a Windows workstation. To process a sample report, you typically create or extract a subset of a complete report. The report in this example was generated on a z/OS system and transferred to the PC as a binary file, and then loaded into the graphical indexer.

The sample data used to create the indexing information must match the actual data to be indexed and loaded into the database. When you load a report into the system, OnDemand uses the indexing parameters, options, and data values that are stored with 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, 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.

Return this portion with your payment.

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


Make check payable to



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

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

TOTAL \$56.97



BOULDER CO 80000-0000

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

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

THANK YOU FOR YOUR PAYMENT

CURRENT CHARGES

DATE DUE
AMOUNT DUE

JAN 29, 1993
\$56.97

SUMMARY OF CURRENT CHARGES

RESIDENCE SERVICE
911 SURCHARGE
CUSTOMER ACCESS SERVICE
WIRING MAINTENANCE PLAN
FEDERAL EXCISE TAX
STATE TAX
LONG DISTANCE CHARGES (ITEMIZED BELOW)

\$25.07
\$0.50
\$3.50
\$0.50
\$0.50
\$0.49
\$26.41

LONG DISTANCE CHARGES

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

PAGE 1

Figure 12. Phone bill

Chapter 2. Using ACIF 23

```

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 13. Phone Bill Data Stream

```

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

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

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

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

```

Figure 14. ACIF Parameters

Key concepts

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

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

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 OnDemand application identifies the type of data that is stored in the system, the method used to index the data, and other information used to load and view reports. This section provides information about the application for the sample phone bill report.

General

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

Assign the application to the application group in which the phone bill report data will be maintained. The application group contains database fields for the account number, customer name, and bill date.

View Information

The View Information page is where you specify information needed by OnDemand client programs to display the phone bills. Some of the information is also used for the indexing parameters.

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

- Code Page. Set the code page to 500. This is the code page of the input data being processed 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 you specify information that is used to generate index data for the report.

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

- CONVERT=N0. The default value for a line data input file. Although the report will be stored in the system as AFP data, you first need to process a sample of the source data with the ACIF graphical indexer. After generating the indexing parameters, change CONVERT to YES.
- CPGID=500. The code page of the input data.
- 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.

Next, define additional ACIF parameters, including those that determine the index data extracted from the report and loaded into the database. To do so, process a sample of the report data with the OnDemand graphical indexer.

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. The client opens the Indexer Properties dialog box. After you click OK or Cancel, the client 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 that are stored with 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 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, define two triggers:

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

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

Defining TRIGGER1

Because the graphical indexer displays the report as it is viewed in OnDemand, the carriage control characters in column one of the data cannot be viewed from the graphical indexer. 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 phone bill report, ACIF should 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. Set the Columns to Search to Carriage Control so that ACIF searches column one of each record. When that is done, the graphical indexer displays the trigger string value (X'F1') in the Value area.

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

Locating the account number

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

The Identifier (Trigger2) determines the name of the trigger parameter. The trigger Type is Group. ACIF should create a group index for each account number that 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. Because the trigger string value can be found in a specific record in each group, only one record will be searched (record 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 that was 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.

Defining fields

ACIF uses fields to determine where to locate index values. For the sample phone bill report, 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. For example, select the account number on the first page in the sample report. Then 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 selected field value.

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. For example, select the customer name on the first page in the sample report. When you select a field value, include sufficient blank columns (to the right of the name) to generate a field length to hold the longest name that ACIF will encounter in an actual report. For example, if the field in an actual report can include values up to 30 characters in length and the sample value is only 17 characters in length, you must select an additional 13 columns in the sample report. After selecting the sample field value, 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 selected field value.

To save the field information, 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. For example, select the billing date on the first page in the sample report. After selecting the sample field value, 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 selected field value.

To save the field information, click OK to add the field and return to the report window.

Defining indexes

The 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 that ACIF creates. For the phone bill report, 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. Accept the suggested default, acct_num. When data is loaded 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 that ACIF generates. For each index that is defined in this example, ACIF should 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. Because ACIF should use the account number index to control the group break, set Break to Yes. The Fields area lists the field parameters that have been defined for the report. 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>> to move 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. Accept the suggested default, name. When the report is loaded 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 that ACIF generates. For all indexes in this example, ACIF should 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. Because ACIF does not use the customer name index to control the group break, set Break to No. The Fields area lists the field parameters that have been defined (in the Fields list) for the report. 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>> to move 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. Accept the suggested default, bill_date. When a report is loaded 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 that ACIF generates. For each index that is defined in this example, ACIF should 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. Because ACIF does not use the billing date index to control the group break, set Break to No. The Fields area lists the field parameters that have been defined for the report. 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>> to move 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.

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, click the Display and Add Parameters icon to return 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 to highlight the area of the report where the field was defined. Click Field1 again to highlight the area on the next page. Click Field2 to highlight 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, 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 information about the format of the input data, the resources required to index the data, the index output file, and the optional user-written programs that may be used to process input, output, and index records and resources. Some of the parameter values are based on the choices that you make on the View Information page. Click Help on each page to display information about the fields. You can also see Chapter 3, "Parameter reference," on page 49 for details.

Click the Data Format tab. Because the line data input should be loaded into the system as AFP data, set Data Conversion to Yes. 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 on the View Information page.

Click the Resource Information tab. Because the input data will be converted to AFP, you must specify values on this page. For the sample phone bill report, specify the name of a form definition and page definition. Because ACIF should collect form definitions, overlays, and page segments, check the appropriate boxes in the Resource File Contents area. Finally, in the Search Paths area, identify where ACIF can find the resources. Enter the names of the directories where the resources reside.

Click OK to save the changes and return to the report window. Close the report window. When prompted, click Yes to save your changes and return to the Indexer Information page.

Defining the application, part 2

View Information

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

The Data Type was initially set to Line to prepare the indexer information. After generating the indexing parameters, setting Convert to Yes (in the Indexer Parameters dialog box) causes the administrative client to automatically set the Data Type to AFP, which is the format of the data as it is stored in the system.

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 that were added based on the settings in the Indexer Properties dialog box.

Load Information

The Load Information page is where you map the index attribute names defined to hold the attribute values ACIF extracts from the report to application group database field names. For the sample phone bill report, the attributes are named the same as the database fields. Therefore, the attributes names do not have to be mapped 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

Click OK to add the application, update the database, and returns to the Administrative Tasks window.

Example three

About the report

This example describes how to create index information for a sample income statement report. An income statement report typically contains hundreds of pages of line data. The report is logically segmented into statements. The beginning of a statement occurs when two conditions exist: a record that contains a skip-to-channel one carriage control and a record that contains the string *Income Statement*. Each statement can contain one or more pages. Figure 15 on page 33 shows an income statement viewed with one of the OnDemand client programs.

For ease of retrieval, the report should be segmented into groups of pages, with one statement in each group. One index row should be created for each group. The row contains three user-defined indexes: the account number, the statement date, and the total income. In addition, page-level indexes will be generated so that users can locate the type of income and the subtotal when they view a statement. The page-level indexes are stored with the document, not in the database (you cannot use page-level indexes to search for documents). ACIF can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters, and by creating an index field with the TYPE=PAGE option. For more information, see the discussion of TYPE=PAGE in “INDEX” on page 67.

Figure 16 on page 34 shows the ACIF indexer parameters required to process the data.

Accessing the sample report

This example provides instructions about using the OnDemand graphical indexer to process a sample report and create indexing information. The graphical indexer is part of the OnDemand administrative client, a program that runs on a Windows workstation. To process a sample report, you typically create or extract a subset of a complete report. The report in this example was generated on a z/OS system and transferred to the PC as a binary file, and then loaded into the graphical indexer.

The sample data used to create the indexing information must match the actual data to be indexed and loaded into the database. When you load a report into the system, OnDemand uses the indexing parameters, options, and data values that are stored with 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, 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.

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 15. Income Statement


```

/* DATA INPUT/OUTPUT CHARACTERISTICS                                */
CC=YES                                                                /* carriage controls present */
CCTYPE=A                                                              /* ANSI controls in EBCDIC  */
CONVERT=YES                                                           /* convert line data input   */
                                                                      /* to AFP so that page-level */
                                                                      /* indexes can be generated   */
CPGID=500                                                             /* code page ID              */
TRC=NO                                                                /* table ref chars not present */
FILEFORMAT=RECORD,133                                                /* Fixed length input file   */

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

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

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

```

Figure 16. 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. Because 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 move to a specific page in the document by using the Go To command. ACIF can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters, and by creating an index field with the TYPE=PAGE option. For more information, see the discussion of TYPE=PAGE in “INDEX” on page 67.

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. To generate page-level indexes that are written to the output file and can be used to move to a specific page in the document, you must convert a line data input file to AFP.

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 OnDemand application identifies the type of data that is stored in the system, the method used to index the data, and other information used to load and view reports. This section provides information about the application for the sample income statement report.

General

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

Assign the application to the application group in which the income statement report data will be maintained. The application group contains database fields for the group indexes: statement number, statement date, and total income. (ACIF will also generate page-level indexes for the types of income and subtotals fields. However, page-level indexes are not stored in the database.)

View Information

The View Information page is where you specify information needed by OnDemand client programs to display the income statements. Some of the information is also used for the indexing parameters.

Even though the income statement report will be stored in the system as AFP data, the Data Type should be set to Line at this time to define the triggers, fields, and

indexes. After defining the indexing information, the Data Type will be reset to AFP, which is the format of the data as it is stored in the system. Other important settings on this page include:

- Code Page. Set the code page to 500. This is the code page of the data as it is viewed by 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 you specify information that is used to generate index data for the report.

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

- CONVERT=NO. The default value for a line data input file. Although the report will be stored in the system as AFP data, a sample of the input line data will be processed by using the graphical indexer. After generating the indexing parameters, CONVERT will be set to YES.
- CPGID=500. The code page of the data as it is viewed by the graphical indexer.
- 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.

Next, define additional ACIF parameters, including those that determine the index data extracted from the report and loaded into the database. To do so, process a sample of the report data with the OnDemand graphical indexer.

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. The client opens the Indexer Properties dialog box. After you click OK or Cancel, the client 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 that are stored with 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, 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.

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, define five triggers:

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

Defining TRIGGER1

Because the graphical indexer displays the report as it is viewed in OnDemand, you 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, ACIF should 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. Set the Columns to Search to Carriage Control so that ACIF searches column one of each record. When that is done, the graphical indexer displays the trigger string value (X'F1') in the Value area.

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

Locating the statement number

To define the trigger used to locate the income statement number, select the string Income Statement in the report. The graphical indexer highlights the string. Click the Trigger icon on the toolbar 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 that is 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. ACIF should 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. Because the total income value can occur in one of two records (depending on the number of address lines in a statement), specify a Record Range. The Start value is the first record that can contain the total income value. In the example, the Start value is the number of the record (7, seven) that contains the trigger string value selected in the report. The End value is the last record that ACIF searches for the trigger. In the 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 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. Because ACIF should collect index values for all the income types found in each group, it does not matter which Type of Income string is selected (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. ACIF should search every record in the group. A float trigger is how this is accomplished. Change the Type to Float. When the Type is changed 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 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. Because ACIF should collect index values for all the subtotals found in each group, it does not matter which Subtotal string is selected (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). ACIF should search every record in the group. A float trigger is how this is

accomplished. Change the Type to Float. When the Type is changed 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 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.

Defining fields

ACIF uses fields to determine where to locate index values. For the sample income statement report, 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. For example, select the statement date on the first page in the sample report. Then 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 the 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 selected field value.

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. For example, select the statement number on the first page in the sample report. Then 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. Because ACIF should locate the statement number field using TRIGGER2, 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 the 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 selected field value.

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. For example, select the total income value on the first page in the sample report. When you select a field value, include two blank columns (to the right of the value) to generate a field length to hold the largest total income value that ACIF will encounter in an actual 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, select an additional two columns in the sample report. After selecting the sample field value, 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. Because ACIF should locate the total income field using TRIGGER3, 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 the 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 selected field value.

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. For example, select one of the type of income values on the first page in the sample report. When you select a field value, include sufficient blank columns (to the right of the value) to generate a field length to hold the largest type of income value that ACIF will encounter in an actual 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, select an additional four columns in the sample report. After selecting the sample field value, 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. Because ACIF should locate the type of income field using TRIGGER4, you 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 the 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 selected field value.

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. For example, select one of the subtotal values on the first page in the sample report. When you select a field value, include sufficient blank columns (to the right of the value) to generate a field length to hold the largest subtotal value that ACIF will encounter in an actual 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, select an additional two columns in the sample report. After selecting the sample field value, 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. Because ACIF should locate the total income field using TRIGGER5, you 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 the 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 selected field value.

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

Defining indexes

The 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 that are stored in the database and the page indexes that are stored with the AFP document, and the type of index ACIF creates. For the income statement report, 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. Accept the suggested default, sdate. When the report is loaded 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. Because ACIF should extract one date index value for each group in the report, 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. Because ACIF should use the date index to control the group break, set Break to Yes. The Fields area lists the field parameters that have been defined (in the Fields list) for the report. 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>> to move 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. Accept the suggested default, incstmt. When data is loaded 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. Because ACIF should extract one income statement value for each group in the report, 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. Because ACIF should use the statement number index to control the group break, set Break to Yes. The Fields area lists the field parameters that have been defined for the report. 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>> to move 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. Accept the suggested default, totinc. When the report is loaded 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. Because ACIF should extract one total income value for each group in the report, 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. Because ACIF should not use the total income index to control the group break, set Break to No. The Fields area lists the field parameters that have been defined for the report. 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>> to move 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. Because ACIF should extract type of income indexes for each page in a statement, 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. 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>> to move 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 Subtotal. The Type of Index determines the type of index ACIF generates. Because ACIF should extract subtotal indexes for each page in a statement, 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. 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>> to move 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.

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, click the Display and Add Parameters icon to return 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 to highlight the area of the report where the field is defined. Click Field1 again to highlight the area on the next page. Click Field2 to highlight 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, 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 information about the format of the input data, the resources required to index the data, the index output file, and the optional user-written programs that may be used to process input, output, and index records and resources. Some of the parameter values are based on the choices that you make on the View Information page. Click Help on each page to display information about the fields. You can also see Chapter 3, "Parameter reference," on page 49 for details.

Click the Data Format tab. Because the input data will be stored in the system as AFP data (to support the page-level indexes in the output file), set Data Conversion to Yes. 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 that were made on the View Information Page.

Click the Resource Information tab. Because the line data input will be converted to AFP, specify the name of a form definition and page definition. To collect form definitions, select the appropriate option in the Resource File Contents area. In the Search Paths area, specify the location of the resources. Enter the name of the directories in which the resources reside.

Click OK to return to the report window. Close the report window. When prompted, click Yes to save your changes and return to the Indexer Information page.

Defining the application, part 2

View Information

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

To define the triggers, fields, and indexes, the Data Type was set to Line, which is the format of the input data. After defining the indexing parameters, the Data Conversion option was set to Yes, causing the Data Type to be set to AFP, which is the format of the data as it is stored in the system.

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 you map index attribute names defined to hold the attribute values that ACIF extracts from the report to application group database field names. For the sample income statement report, the attribute names are the same as the database fields. Therefore, there is no 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 the index data is loaded into 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

Click OK to add the application, update the database, and return to the Administrative Tasks window.

Example four

About the report

This example describes how to create indexing information for an AFP input 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 move to a specific page in 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 17 on page 45 shows a page of a sample document viewed with one of the OnDemand client programs.

For faster retrieval, the input data should be loaded into the system as OnDemand large objects in groups of 20 pages. Figure 18 on page 46 shows the indexer parameters required for ACIF to process the AFP file.

Accessing the report

Because there is no need to specify TRIGGER, FIELD, and INDEX parameters for the report, a sample of the input does not have to be processed 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 17. AFP document

```

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

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

/* INDEXING INFORMATION                                           */
DCFPGENAMES=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=/usr/lpp/ars/pubs/reslib                                  /* resource library          */
RESTYPE=NONE                                                       /* do not collect resources  */

```

Figure 18. 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 the input data into groups of pages, compressed inside of a large object. You specify the number of pages in a group. When the user selects a document for viewing, the client retrieves and uncompresses the first group of pages. As the user moves from page to page of the document, the client automatically retrieves and uncompresses the appropriate groups of pages. This type of page-level information is stored in the index file. ACIF can generate this type of page-level information whether or not the input data is being converted to AFP. This type of page-level information is essential for loading OnDemand large objects. This type of page-level information is generated by specifying the INDEXOBJ=ALL parameter.

Page Identifiers. For AFP data, identifies each page in a report and provides another way to move to a specific page in a report. Typically extracted from page-level TLEs contained in the document. This type of page-level information is stored in the output file. ACIF can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters. In this example, the input data already contains the page-level TLE information so a TYPE=PAGE index does not have to be specified.

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 OnDemand application identifies the type of data that is stored in the system, the method used to index the data, and other information used to load and view reports. This section provides information about the application for the sample AFP file.

General

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

Assign the application to the application group in which the input data will be maintained. The application group contains database fields for the document date and document number.

View Information

The View Information page is where you specify information needed by OnDemand client programs to display data retrieved from the system. Some of the information is also used for the indexing parameters.

Because the data will be stored in OnDemand as AFP data, set the Data Type to AFP.

Indexer Information

The Indexer Information page is where you specify ACIF as the Indexer and create additional ACIF parameters, including those that identify the format of the AFP file, the type of indexes that ACIF generates, and the resources required by ACIF to process the input file. In this example, the parameters will be created by using the keyboard option on the Indexer Information page.

Creating indexing parameters

In the Add an Application window, click the Indexer Information page. In the Parameter Source area, select Keyboard. Then click Modify to open the Edit Indexer Parameters window, a standard edit window where you can type, modify, and delete ACIF parameters.

Note: The system does not verify the parameters, options, or data values that you type in the Edit Indexer Parameters window.

Defining the data format

The following parameters describe the format of the AFP file. Note that whenever you process input data with ACIF and store AFP data in the system, you must specify CONVERT=YES.

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

Defining indexing information

The following parameter describes the type of index information that ACIF generates. This parameter causes ACIF to generate page-level indexes in addition to group-level indexes. In this example, two types of page-level information will be generated: page-level indexes in the index file to support OnDemand large objects and page-level indexes in the output file that can be used from the client to move to a specific page in the document.

- INDEXOBJ=ALL

The following parameter is required because the sample AFP file was generated by the DCF application, and you must change the value of the DCFPAGENAMES parameter.

- DCFPAGENAMES=YES

Defining resource information

The following parameters are required so that ACIF can process the AFP file. These are the minimum parameters required by ACIF to process AFP data. The parameters specify the name of a standard form definition, the directory that contains resources, and that ACIF should not collect resources (in this example, the AFP file contains all of the required resources).

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

Close the Edit Indexer Parameters window. When prompted, click Yes to save your changes, update the database, and 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 file. Use the scroll bars to review the parameters.

Load Information

The sample AFP file contains TLEs that contain the index attribute names and values. ACIF extracts the index attribute names and values from the input file and writes them to an index object file. The index attribute names in the input file must match the application group database field names. If they do not, then you must 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 that may be used when loading index data into 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

Click OK to add the application, update the database, and return to the Administrative Tasks window.

Chapter 3. Parameter reference

This parameter reference assumes that you will run ACIF on an OnDemand UNIX or Windows server. If you run ACIF on a z/OSsystem, then the defaults for certain parameters and values change from ASCII to EBCDIC. Refer to the CCTYPE (page 49) and CPGID (page 52) parameters, the MASK subparameter of the TRIGGER parameter (page 89), and the USERMASK parameter (page 95) for details.

This parameter reference also assumes that you will use the ARSLOAD program to index and load your reports. When you use the ARSLOAD program to process your reports, it automatically invokes ACIF if the input data needs to be indexed. The ARSLOAD program ignores the INDEXDD, INPUTDD, MSGDD, OUTPUTDD, PARMDD, and RESOBJDD parameters, if specified. If you run ACIF from the command prompt or if you run ACIF on a z/OS system, then verify the values of the INDEXDD (page 70), INPUTDD (page 74), MSGDD (page 76), OUTPUTDD (page 78), PARMDD (page 80), and RESOBJDD (page 85) 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

Z

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 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, the CHARS parameter also provides the names of the fonts ACIF saves in the resource group. The CHARS command can also be used to specify the font used for the entire report when the input data does not contain TRCs and the specified page definition does not name a font.

Syntax

CHARS=*fontlist*

Options and values

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

CHARS=GT10,GT12,GT24

The font name is limited to four alphanumeric or national characters and should not include the two-character prefix of the coded font name (X0 through XG). For example, the coded font X0GT10 would be specified as GT10. On UNIX servers, the font name is case sensitive.

Related parameters

PAGEDEF parameter on page 79.

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. Resources are not collected when you specify CONVERT=NO.

To generate page-level information in the output file you must specify CONVERT=YES. This type of page-level information is used in the client to move to specific pages in a document. ACIF can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters, and by creating an index field with the TYPE=PAGE option. For more information, see the discussion of TYPE=PAGE in “INDEX” on page 67.

To generate page-level information in the index file, you do not have to convert the input data. ACIF can generate this type of page-level information whether or not the input data is being converted to AFP. This type of page-level information is essential for loading OnDemand large objects. This type of page-level information is generated by specifying the INDEXOBJ=ALL parameter. Therefore, if you do not

need to convert the input line data to AFP, but you do want ACIF to generate this type of page-level information for large objects, 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 86.

CPGID

Required

No

Default Value

850 (ASCII)

500 (EBCDIC)

Data Type

AFP, Line

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

Syntax

CPGID=*value*

Options and values

The *value* can be:

850

The default IBM code page.

code page identifier

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

DCFPAGENAMES

Required

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, Line

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.

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).

EMPTYOK

If indexing is requested by specifying the TRIGGER, FIELD, and INDEX parameters, ACIF must find a group indexing field before the page specified by the INDEXSTARTBY parameter. Under normal processing, if ACIF fails to find a group indexing field, ACIF issues error message APK448S/0425-448 and ends with RC 16. When EMPTYOK is specified and a group indexing field is not found, ACIF will issue message 422 with RC 64 and then issue message 440 and RC 0.

Note: If no indexing information is found, the file does not load into Content Manager OnDemand.

FRACLINE

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

IDXCPGID

Specify this option for Unicode documents. ACIF adds extensive code page information to the AFP document (if ACIF is converting the document to AFP) and to the index file, so that the document can load correctly into Content Manager OnDemand and display properly. This parameter does not affect printing.

Important:

1. Only the following four Unicode code pages are supported:
 - 1200** UTF-16 BE (previously UCS-2)
 - 1208** UTF-8
 - 13488** UTF-16 BE
 - 17584** UTF-16 BE
2. Ensure that you indicate the code page of the document and the extracted index values by using the CPGID parameter. Ensure that all the extracted index values are in the same code page.
3. Ensure that you express the trigger and index names in the TRIGGER and INDEX parameters in the code page that is specified by the CPGID parameter.
4. Ensure that you express the trigger and index names in Big Endian.
5. Ensure that you extract the field values from the document in Big Endian format.
6. IDXCPGID can be used with both CONVERT=YES and CONVERT=NO.
7. Do not use a mask on the FIELD parameter when you use IDXCPGID.
8. Do not use IDXCPGID if the input is AFP or mixed-mode.

This example contains sample ACIF parameters for a code page 1200 (UCS-2) document:

```
CC=YES
CCTYPE=A
CPGID=1200
FILEFORMAT=RECORD,401
TRIGGER1=*,228,X'0050004100470045',(TYPE=GROUP)          /* P A G E */
FIELD1=0,246,10,(TRIGGER=1,BASE=0)
FIELD2=0,-76,16,(TRIGGER=1,BASE=TRIGGER)
INDEX1=X'0070006100670065',FIELD1,(TYPE=GROUP,BREAK=YES) /* page */
INDEX2=X'006E0061006D0065',FIELD2,(TYPE=GROUP,BREAK=YES) /* name */
```

```
EXTENSIONS=IDXCPGID
FORMDEF=F1IBMTU3
PAGEDEF=P1IBMTU3
RESLIB=\acif\reslib2
```

The above example illustrates these important points:

- The trigger and index names are expressed in Big Endian UCS-2. The trigger and index names must be in the code page given by the CPGID parameter.
- The field values must be extracted from the document in Big Endian format. In the example, on the first page, the following 10 bytes are extracted for FIELD1:

```
X'00200020002000200031'    /*    1 */
```

and the following 16 bytes are extracted for FIELD2:

```
X'002000500045004C0053004800320032'    /* PELSH22 */
```

PRCOLOR

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

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.

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 66.
CPGID parameter on 52.

FDEFLIB

Required

No

Default Value

(None)

Data Type

AFP

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

1. The paths you specified with the USERLIB parameter, if any.

2. The paths you specified with the FDEFLIB parameter, if any.
3. The paths you specified with RESLIB parameter, if any.
4. On UNIX servers, the paths specified on the PSFPATH environment variable (if it is set). On Windows servers, ACIF first attempts to get the path from the registry; if that fails, ACIF attempts to get the path from the PSFPATH environment variable.
5. On AIX® servers, the directory /usr/lpp/psf/reslib (if it exists). On HP-UX and Solaris servers, the directory /opt/psf/reslib (if it exists).

Syntax

FDEFLIB=*pathlist*

Options and values

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

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

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

Important: The total number of all characters in the string of path names cannot exceed 4095 bytes.

Related parameters

RESLIB parameter on page 84.

USERLIB parameter on page 94.

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 and that contains one or more columns of sorted data. Because it is not always practical to store every index value in the database, ACIF extracts the first and last sorted values in each group. Depending on the format (ASCII or EBCDIC) of the data, the data is sorted according to the collating sequence of the code page.
- Mask field, which must be based on a floating trigger and which uses a mask to match data located in the field columns.

Trigger field syntax

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

Options and values

n

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

record

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

column

The relative column number from the BASE. This is the column number where ACIF begins to search for the field. A value of 1 (one) refers to the first byte in the record. For files containing carriage-control characters, column one refers to the carriage-control. For those applications that use a specific carriage-control character to define page boundaries (for example, skip-to-channel one), consider defining the value of the carriage-control character as one of the TRIGGER parameters. If you specify BASE=0, the *column* value can be 1 to 32756. If you specify BASE=TRIGGER, the *column* value can be -32756 to 32756.

Note: When specifying the column number of the field, if the specified value exceeds the physical length of the record, ACIF reports an error condition and terminates processing, unless you specify a DEFAULT value.

length

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

TRIGGER=*n*

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

BASE={0 | TRIGGER}

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

DEFAULT='value'

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

The DEFAULT keyword supports the use of Download, which can be used to transmit data from the JES Spool to the OnDemand server. JES, by default, will truncate trailing blanks at the end of records. For example, assume that a report contains fixed length records, each 80 bytes in length. Columns 77 through 80 of the records contain audit data that is generated by a user-defined program. If a record has not been audited, the columns contain blanks. During file transmission, JES eliminates the blanks from the end of the records. When processing the records that have been truncated, ACIF will fail unless you specify a DEFAULT value. For more information regarding problems that can occur when using Download, see OnDemand support on the Web at <http://www.ibm.com/software/data/ondemand/mp/support.html> and search for document number 1005882.

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. Specify BASE=0 because the field data always starts in the same column.

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

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

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

Constant field syntax

FIELD*n*=*constant*

A *constant field* cannot be concatenated in an index 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.

Options and values

n

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

constant

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

Examples

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

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

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

```
FIELD3=X'F0F0F0F060'  
FIELD4=0,66,14  
INDEX3=X'818383A36D95A494',FIELD3,FIELD4,(TYPE=GROUP,BREAK=YES)
```

Transaction field syntax

```
FIELDn=*,*,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 95) 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)

MASK field syntax

The MASK keyword of the FIELD parameter now supports a field that is based on a floating trigger. Previously, the MASK keyword could only be used with a transaction field. **Important:** 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.

Use the following syntax to specify a field with a mask when the field is based on a floating trigger:

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

Where:

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 from ± 0 to 255.

column

The relative column number from the BASE (specified with the BASE keyword of the FIELD parameter). 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, then the *column* value can be from 1 to 32756. If you specify BASE=TRIGGER, then the *column* value can be from -32756 to 32756. If the specified value exceeds the physical length of the record, then ACIF reports an error condition and terminates processing.

length

The number of contiguous bytes (characters) that compose the field. The supported range of values are from 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, then 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. When using a MASK, you must specify a trigger that was defined with TYPE=FLOAT.

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 you specify BASE=0, then ACIF adds zero to the field column offset. If you specify BASE=TRIGGER, then ACIF adds the starting column number of the trigger string value to the field column offset. You should specify BASE=0 if the field data always starts in a specific column. You should specify 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; you would specify `BASE=TRIGGER` and a column offset of 10 (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. 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='####',DEFAULT=X'F1F0F0F0')
```

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

DEFAULT=*value*

Optionally use to specify a default index value when a record is not long enough to contain the field data. (However, if a record is not long enough and you do not specify a default value, ACIF will fail.) The default value can be specified as either character data or hexadecimal data. If the data to be specified is anything other than ASCII, then you must specify the default value as a hexadecimal value by using the format `X'constant'`, where constant is hexadecimal data (for example, `X'F1F0F0F0'`). The default value can be from 1 (one) to 250 characters in length. **Note:** The MASK is not applied to the default value.

Related parameters

CPGID parameter on page 52.

INDEX parameter on page 67.

TRIGGER parameter on page 89.

FILEFORMAT

Required

No

Default Value

STREAM

Data Type

AFP, Line

Identifies the format of the input file, and optionally, the character or characters that separate records in the input file.

Syntax

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

Options and values

The values are:

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

The input file has no length information; it is a stream of data separated by a newline character. Files with STREAM format typically come from a workstation operating system such as AIX, HP-UX, Sun Solaris, and Windows. The NEWLINE keyword identifies the hexadecimal character or characters that delimit records in the data stream. The NEWLINE keyword supports a two-character line delimiter, which is common in data from DOS and Windows. The following example shows how to specify the FILEFORMAT parameter to process input data that contains two-character line delimiters:

FILEFORMAT=STREAM, (NEWLINE=X'0D0A')

If the NEWLINE keyword is not specified, the default line delimiter for ASCII data is X'0A' and the default line delimiter for EBCDIC data is X'25'.

RECORD[,n]

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

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

FONTLIB

Required

No

Default Value

(None)

Data Type

AFP

Required	Default Value	Data Type
No		AFP

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

1. The paths you specified with the USERLIB parameter, if any.
2. The paths you specified with the FONTLIB parameter, if any.
3. The paths you specified with the RESLIB parameter, if any.
4. On UNIX servers, the paths specified in the PSFPATH environment variable (if it is set). On Windows servers, ACIF first attempts to get the path from the registry; if that fails, ACIF attempts to get the path from the PSFPATH environment variable.
5. On AIX servers, the directory /usr/lpp/psf/reslib, if it exists. On HP-UX and Solaris servers, the directory /opt/psf/reslib (if it exists).

6. On AIX servers, the directory `/usr/lpp/afpfonts`, if it exists. On HP-UX and Solaris servers, the directory `/opt/psf/afpfonts` (if it exists).
7. On AIX servers, the directory `/usr/lpp/psf/fontlib`, if it exists. On HP-UX and Solaris servers, the directory `/opt/psf/fontlib` (if it exists).

Syntax

`FONTLIB=`*pathlist*

Options and values

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

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

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

Important: The total number of all characters in the string of path names cannot exceed 4095 bytes.

Related parameters

RESLIB parameter on page 84.

USERLIB parameter on page 94.

FORMDEF

Required

Yes

Default Value

(None)

Data Type

AFP

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

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

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

- On UNIX and Windows servers, specify FILEFORMAT=RECORD,*n* (where *n* is the logical record length defined for the file).
- In z/OS, specify record format FBA (fixed block with ANSI carriage control characters or FBM (fixed block with machine carriage control characters).
- If the length of the records in the form definition is greater than the logical record length defined for the file, you must specify variable length records:
 - On UNIX and Windows servers, specify FILEFORMAT=RECORD. The first two bytes of each record determine the record length.
 - In z/OS, specify record format VBA (variable blocked with ANSI carriage control characters or VBM (variable blocked with machine carriage control characters).

Syntax

FORMDEF=*fdefname*

Options and values

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

Related parameters

FDEFLIB parameter on page 55.

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 and 9999.

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

When indexing transaction data with a GROUPRANGE index, you typically set the GROUPMAXPAGES parameter to control the maximum number of pages in a group.

Syntax

GROUPMAXPAGES=*value*

Options and values

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

Related parameters

INDEX parameter on page 67.

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, a FIELD cannot be 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 67.

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. IBM recommends 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. You must define at least one index parameter. You can define up to 32 index parameters. When you define a group index, IBM recommends that you 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 move to a specific page in the document by using the Go To command in the client.

To generate page-level information in the output file you must specify CONVERT=YES. This type of page-level information is used in the client to move to specific pages in a document. ACIF can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters, and by creating an index field with the TYPE=PAGE option.

Syntax

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

Options and values

n

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

name

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

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

FIELD*nn*

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

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

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

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

TYPE=*type*

The type of index ACIF generates. You can define group indexes for AFP and line data. You can define page indexes for AFP and line 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

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 a document, you can use the page indexes to move to a specific page in the document by using the Go To command in the client.

To generate page-level information, you must specify CONVERT=YES. This type of page-level information is used in the client to move to specific pages in a document. ACIF can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters, and by creating an index field with the TYPE=PAGE option. When you define a PAGE index, you must specify INDEXOBJ=ALL; otherwise, ACIF will not write the page index data to the index object file.

TYPE=PAGERANGE,BREAK=NO

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 a document, you can use the page indexes to move to a specific page in the document with the Go To command in the client.

To generate page-level information, you must specify CONVERT=YES. This type of page-level information is used in the client to move to specific pages in a document. ACIF can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters, and by creating an index field with the TYPE=PAGERANGE option. When you define a PAGERANGE index, you must specify INDEXOBJ=ALL; otherwise, ACIF will not write the pagerange index data to the index object file.

Examples

Group index

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

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

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

```
INDEX1='95819485',FIELD1,(TYPE=GROUP,BREAK=YES)
INDEX2='818383A46D95A494',FIELD2,(TYPE=GROUP,BREAK=NO)
```

Grouprange index

The following index parameter causes ACIF to generate grouprange indexes for loan number index values. ACIF extracts the beginning and ending loan numbers in each group of pages. The input data is encoded in EBCDIC. A grouprange index must be based on a transaction field. Because a grouprange index cannot be used to break a report into groups of page, the GROUPMAXPAGES parameter can be used to determine the number of pages in a group. ACIF closes the current group and begins a new group when the number of pages in the group is equal to the value of the GROUPMAXPAGES parameter.

```
INDEX2='4C6F616E204E756D626572',FIELD2,(TYPE=GROUPRANGE,BREAK=NO)
GROUPMAXPAGES=100
```

Page index

The following index parameter causes ACIF to generate page indexes for subtotal values (the attribute name that appears in the Go To dialog box is Subtotal). The input data 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 56.

INDEXOBJ parameter on page 71.

INDEXDD

Note: This parameter is ignored when you process reports with the ARSLOAD program.

Required

No

Default Value

INDEX

Data Type

AFP, Line

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

SyntaxINDEXDD=*filename***Options and values**

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

INDEXOBJ**Required**

No

Default Value

GROUP

Data Type

AFP, Line

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

SyntaxINDEXOBJ=*value***Options and values**

The *value* can be:

GROUP

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

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

ALL

ACIF includes group-level and page-level indexes in the index object file.

You must specify INDEXOBJ=ALL for reports that require page-level index support. There are two types of page-level index information, and different ways to generate the information.

- Page-level information in the index file. ACIF can generate this type of page-level information whether or not the input data is being converted to AFP. This type of page-level information is essential for loading OnDemand large objects. This type of page-level information is generated by specifying the INDEXOBJ=ALL parameter.
- Page-level information in the output file. This type of page-level information is used in the client to move to specific pages in a document. ACIF can only generate this type of page-level information when converting the input data to AFP. This type of page-level information is generated by specifying the CONVERT=YES and INDEXOBJ=ALL parameters, and by creating an index

field with the TYPE=PAGE option. For more information, see the discussion of TYPE=PAGE in “INDEX” on page 67.

NONE

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

BDTLY

The INDEXOBJ parameter now includes support for stapling on document boundaries when processing for Infoprint Manager. **Note:** This function should not be used with OnDemand.

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 it's 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.

To enable BDT/EDT pass through, specify the BDTLY option on the INDEXOBJ parameter. For example:

```
INDEXOBJ=BDTLY
```

Related parameters

INDEX parameter on page 67.

INDEXSTARTBY

Required

No

Default Value

1

Data Type

AFP, Line

Determines the page number by which ACIF must find a group indexing field. A group indexing field is a field which is based on a group or recordrange trigger. ACIF fails if it does not find a group 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.

When you use INDEXSTARTBY to skip header pages, ACIF does not copy the non-indexed pages to the output file. For example, if you specify INDEXSTARTBY=3, ACIF finds the first index on page three, and ACIF skips pages one and two. Page three will be the first page in the output file.

Important: GROUP, RECORDRANGE, and FLOAT triggers apply only if you use enhanced indexing. A group indexing field is based on a GROUP or RECORDRANGE trigger, not on a FLOAT trigger.

Syntax

INDEXSTARTBY=*value*

Options and values

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

- 1 Specifies that ACIF must find a group index on the first page.
- nn* Specifies the output page number (0–99) by which ACIF must find the group index criteria specified. 0 indicates that there is no limit to the page where ACIF must find a group indexing field.

This parameter is helpful if your file contains pages that you want the indexer to skip, for example, header pages or alignment pages. If your file contains two header pages, you can specify a page number one greater than the number of header pages (INDEXSTARTBY=3).

If ACIF does not find a group indexing field before the page number that is specified in the INDEXSTARTBY parameter, it issues a message and stops processing.

INDEXEXIT

Required

No

Default Value

(None)

Data Type

AFP, Line

Identifies the name or the full path name of the index record exit program. This is the program ACIF calls for every record (line or structured field) it writes in the index object file. For more information about optional program exits you can use to customize how ACIF handles input and output data, please refer to Chapter 5, “User exits and attributes of the input file,” on page 99.

Syntax

INDEXEXIT=*name*

Options and values

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

INPEXIT

Required

No

Default Value

(None)

Data Type

AFP, Line

Identifies the name or the full path name of the input record exit program. This is the program ACIF calls for every record (line) it reads from the input file. For more information about optional program exits you can use to customize how ACIF handles input and output data, please refer to Chapter 5, “User exits and attributes of the input file,” on page 99.

SyntaxINPEXIT=*name***Options and values**

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

INPUTDD

Note: This parameter is ignored when you process reports with the ARSLOAD program.

Required

No

Default Value

stdin

Data Type

AFP, Line

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

SyntaxINPUTDD=*filename***Options and values**

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

INSERTIMM**Required**

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 IMM into the output data.

YES

ACIF inserts IMM 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 64.

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. The maximum value for LINECNT is 999.

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 86.

MSGDD

Note: This parameter is ignored when you process reports with the ARSLOAD program.

Required

No

Default Value

stderr

Data Type

AFP, Line

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

Syntax

MSGDD=*filename*

Options and values

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

NEWPAGE

Required

No

Default Value

1

Data Type

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). For example, the numbers 1 to 12 would correspond to the values x'31' - x'39' and x'41' - x'43 in the data, if the data were encoded in ASCII.

Related parameters

The CC parameter on page 49.

The CONVERT parameter on page 51.

OUTEXIT

Required

No

Default Value

(None)

Data Type

AFP, Line

Identifies the name or the full path name of the output record exit program. ACIF calls this program for every output record (every line or structured field) it writes to the output file. For more information about optional program exits you can use to customize how ACIF handles input and output data, please refer to Chapter 5, "User exits and attributes of the input file," on page 99.

Syntax

OUTEXIT=*name*

Options and values

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

OUTPUTDD

Note: This parameter is ignored when you process reports with the ARSLOAD program.

Required

No

Default Value

stdout

Data Type

AFP, Line

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

Syntax

OUTPUTDD=*name*

Options and values

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

OVLYLIB

Required

No

Default Value

(None)

Data Type

AFP

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

1. The paths you specified with USERLIB, if any.
2. The paths you specified with OVLYLIB, if any.
3. The paths you specified with the RESLIB parameter, if any.
4. On UNIX servers, the paths specified in the PSFPATH environment variable (if it is set). On Windows servers, ACIF first attempts to get the path from the registry; if that fails, ACIF attempts to get the path from the PSFPATH environment variable.
5. On AIX servers, the directory /usr/lpp/psf/reslib (if it exists). On HP-UX and Solaris servers, the directory /opt/psf/reslib (if it exists).

Syntax

OVLYLIB=*pathlist*

Options and values

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

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

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

Important: The total number of all characters in the string of path names cannot exceed 4095 bytes.

Related parameters

RESLIB parameter on page 84.

USERLIB parameter on page 94.

PAGEDEF

Required

No

Default Value

(None)

Data Type

AFP

Identifies the file name of the page definition. A page definition defines the page format that ACIF uses to compose the input file into pages. ACIF requires a page definition to convert an input file that contains S/390[®] line data, mixed-mode data, or unformatted ASCII data into AFP.

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

- If you specify the PAGEDEF parameter, the name of the inline page definition must match the name of the specified page definition, or you must specify PAGEDEF=DUMMY. If the name specified on the PAGEDEF parameter does not match the name of an inline page definition, ACIF looks for the page definition in the PDEFLIB search path. If you specify PAGEDEF=DUMMY and there is no inline page definition, ACIF searches the PDEFLIB search path for a page definition named DUMMY. ACIF reports an error and stops if unable to locate the page definition.
- If a page definition resource is included inline with the data, the file must be identified as containing carriage control characters (you must specify CC=YES).
- If the length of the records in the page definition is less than or equal to the logical record length defined for the file, you can specify fixed length records for the record format:
 - On UNIX and Windows servers, specify FILEFORMAT=RECORD,*n* (where *n* is the logical record length defined for the file).
 - In z/OS, specify record format FBA (fixed block with ANSI carriage control characters or FBM (fixed block with machine carriage control characters).

- If the length of the records in the page definition is greater than the logical record length defined for the file, you must specify variable length records:
 - On UNIX and Windows servers, specify FILEFORMAT=RECORD. The first two bytes of each record determine the record length.
 - In z/OS, specify record format VBA (variable blocked with ANSI carriage control characters or VBM (variable blocked with machine carriage control characters).

Important: Inline page definitions are removed from the output data, even if you specify RESTYPE=INLINE or RESTYPE=INLONLY. Page definitions are not saved in the output resource library.

Syntax

PAGEDEF=*pdefname*

Options and values

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

Related parameters

PDEFLIB parameter on page 81.

PARMDD

Note: This parameter is ignored when you process reports with the ARSLOAD program.

Required

No

Default Value

(None)

Data Type

AFP, Line

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

Syntax

PARMDD=*filename*

Options and values

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

PDEFLIB

Required

No

Default Value

(None)

Data Type

AFP

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

1. The paths you specified with the USERLIB parameter, if any.
2. The paths you specified with the PDEFLIB parameter, if any.
3. The paths you specified in the RESLIB parameter, if any.
4. On UNIX servers, the paths specified in the PSFPATH environment variable (if it is set). On Windows servers, ACIF first attempts to get the path from the registry; if that fails, ACIF attempts to get the path from the PSFPATH environment variable.
5. On AIX servers, the directory /usr/lpp/psf/reslib (if it exists). On HP-UX and Solaris servers, the directory /opt/psf/reslib (if it exists).

Syntax

PDEFLIB=*pathlist*

Options and values

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

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

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

Important: The total number of all characters in the string of path names cannot exceed 4095 bytes.

Related parameters

PAGEDEF parameter on page 79.

RESLIB parameter on page 84.

USERLIB parameter on page 94.

PRMODE

Required

No

Default Value

(None)

Data Type

Line, SCS, and Global DJDE

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*

The PRMODE parameter also supports specifying an eight-byte alphanumeric string. The value is supplied to all of the ACIF user exits. Usage: PRMODE=aaaaaaaa, where aaaaaaaaa is the alphanumeric string.

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

SOSI4 is 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

Any eight-byte alphanumeric string. This value is supplied to all of the ACIF user exits. Using the AFPDS value indicates that the data contains MO:DCA-P structured fields.

Related parameters

The CHARS parameter on page 50.

PSEGLIB

Required

No

Default Value
(None)

Data Type
AFP

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

1. The paths you specified with the USERLIB parameter, if any.
2. The paths you specified with the PSEGLIB parameter, if any.
3. The paths you specified with the RESLIB parameter, if any.
4. On UNIX servers, the paths specified in the PSFPATH environment variable (if it is set). On Windows servers, ACIF first attempts to get the path from the registry; if that fails, ACIF attempts to get the path from the PSFPATH environment variable.
5. On AIX servers, the directory /usr/lpp/psf/reslib (if it exists). On HP-UX and Solaris servers, the directory /opt/psf/reslib (if it exists).

Syntax

PSEGLIB=*pathlist*

Options and values

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

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

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

Important: The total number of all characters in the string of path names cannot exceed 4095 bytes.

Related parameters

RESLIB parameter on page 84.
USERLIB parameter on page 94.

RESEXIT

Required
No

Default Value
(None)

Data Type
AFP

Identifies the name or the full path name of the resource exit program. This is the program ACIF calls each time it attempts to retrieve a requested resource from a directory. For more information about optional program exits you can use to customize how ACIF handles input and output data, please refer to Chapter 5, "User exits and attributes of the input file," on page 99.

Syntax

RESEXIT=*name*

Options and values

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

RESFILE

Note: This parameter is valid only on z/OS systems.

Required

No

Default Value

SEQ

Data Type

AFP

Determines the format of the resource file (z/OS systems) 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.

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

Syntax

RESFILE=*type*

Options and values

The *type* can be:

SEQ

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

PDS

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

RESLIB

Required

No

Default Value

(None)

Data Type

AFP

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

definitions, form definitions, bar code objects, image objects, or graphics objects). ACIF searches for resources in the following order:

1. Paths you specified with the USERLIB parameter, if any.
2. Paths you specified with the FDEFLIB, FONTLIB, PDEFLIB, PSEGLIB, and OVLYLIB parameters, if any, for specific types of resources.
3. Paths you specified with the RESLIB parameter, if any.
4. On UNIX servers, the paths specified in the PSFPATH environment variable (if it is set). On Windows servers, ACIF first attempts to get the path from the registry; if that fails, ACIF attempts to get the path from the PSFPATH environment variable.
5. On AIX servers, the directory /usr/lpp/psf/reslib, if it exists. On HP-UX and Solaris servers, the directory /opt/psf/reslib (if it exists).
6. For fonts, on AIX servers, the directory /usr/lpp/afpfonts, if it exists. On HP-UX and Solaris servers, the directory /opt/psf/afpfonts (if it exists).
7. For fonts, on AIX servers, the directory /usr/lpp/psf/fontlib, if it exists. On HP-UX and Solaris servers, the directory /opt/psf/fontlib (if it exists).

Syntax

RESLIB=*pathlist*

Options and values

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

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

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

Important: The total number of all characters in the string of path names cannot exceed 4095 bytes.

Related parameters

FONTLIB parameter on page 63.

FDEFLIB parameter on page 55.

OVLYLIB parameter on page 78.

PDEFLIB parameter on page 81.

PSEGLIB parameter on page 82.

USERLIB parameter on page 94.

RESOBJDD

Note: This parameter is ignored when you process reports with the ARSLOAD program.

Required

No

Default Value

RESOBJ

Data Type

AFP

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

Syntax

RESOBJDD=*filename*

Options and values

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

RESTYPE

Required

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.

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 Content Manager OnDemand. Content Manager OnDemand requires a separate resource file.

INLONLY

Specifies that inline resources are written to the output file. ACIF will look only inline for the resources. External libraries will not be searched. A resource file will not be created.

Important: Do not use the INLONLY option for documents that are 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, for example:

- 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 76.

The RESLIB parameter on page 84.

The RESOBJDD parameter on page 85.

TRACE

Note: This parameter is valid only on z/OS systems.

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

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.

Consider the following when you use TRCs:

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

For more information about TRCs, see *Advanced Function Presentation: Programming Guide and Line Data Reference*.

Syntax

TRC=*value*

Options and values

The *value* can be:

NO

The input does not contain TRCs.

YES

The input does contain TRCs.

Related parameters

The CHARS parameter on page 50.

TRIGGER

This parameter should not be used if the document contains Tagged Logical Element (TLE) structured fields. ACIF will issue an error message if the TRIGGER parameter is used when the document contains TLE structured fields.

Required

Yes

Default Value

(None)

Data Type

AFP, Line

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

When running ACIF on a Windows server and you define a TRIGGER parameter using structured field data, you must change the order of the length bytes in the trigger string value. The length bytes are bytes two and three of a structured field. Some instructions in the x86 architecture use the length bytes in the reverse of the order that they appear in the input data. ACIF automatically changes the order of the length bytes in all input structured fields before indexing the data. For example, in the input data, a structured field may appear as follows:

```
5A0010D3EEEE00. . .
```

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

```
5A1000D3EEEE00. . .
```

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

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

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

Syntax

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

Options and values

n

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

record

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

column

The beginning column where ACIF locates the trigger string value. The supported range of column numbers is 0 to 32756. To force ACIF to scan every record from left to right for the trigger string value, specify an * (asterisk) or 0 (zero) for the column. A 1 (one) refers to byte one of the record. Alternatively, you can specify a beginning and ending column range and separate them by a

colon. If you specify a column range, the beginning column cannot be zero, and the ending column must be greater than the beginning column. See the example below.

Important: Scanning every record can incur a substantial performance penalty. The overhead required to scan every record can cause the indexing step of the load process to take considerably longer than normal. Whenever possible, specify a beginning column number.

value

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

TYPE=*type*

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

GROUP

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

GROUP,RECORDRANGE=(*start,end*)

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

FLOAT

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

About group triggers

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.

Notes

1. ACIF requires that at least one group TRIGGERn value appear within the page range that is specified by the INDEXSTARTBY parameter. If no group TRIGGERn 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'D7C1C7C54040F1',(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=6,50:52,X'C1838396A495A340D5A494828599',(TYPE=GROUP)
```

Group trigger with column range

The following trigger parameter causes ACIF to attempt to match the string value Account Number beginning in columns fifty, fifty-one, or fifty-two 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=6,50:52,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)
```

Trigger using structured field data

Note: This example is for ACIF running on a Windows server

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

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

Related parameters

The FIELD parameter on page 56.

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 52.

USERLIB

Required

No

Default Value

(None)

Data Type

AFP

Identifies the names of user directories containing AFP resources for processing the input file. The directories can contain any AFP resources (fonts, page segments, overlays, page definitions, form definitions, bar code objects, image objects, or graphics objects). By convention, these resources are typically used by one user, as opposed to the system resources (specified with the RESLIB parameter) that are shared by many users. Therefore, you should use the USERLIB parameter to specify resources that are not retrieved with the FDEFLIB, FONTLIB, OVLYLIB, PDEFLIB, or PSEGLIB parameters. ACIF searches for resources in the following order:

1. Paths you specify with the USERLIB parameter, if any.
2. Paths you specify with the FDEFLIB, FONTLIB, OVLYLIB, PDEFLIB, or PSEGLIB parameters, for specific types of resources, if any.
3. Paths you specify with the RESLIB parameter, if any.
4. On UNIX servers, the paths specified in the PSFPATH environment variable (if it is set). On Windows servers, ACIF first attempts to get the path from the registry; if that fails, ACIF attempts to get the path from the PSFPATH environment variable.
5. On AIX servers, the directory /usr/lpp/psf/reslib, if it exists. On HP-UX and Solaris servers, the directory /opt/psf/reslib (if it exists).
6. For fonts, on AIX servers, the directory /usr/lpp/afpfonts, if it exists. On HP-UX and Solaris servers, the directory /opt/psf/afpfonts (if it exists).
7. For fonts, on AIX servers, the directory /usr/lpp/psf/fontlib, if it exists. On HP-UX and Solaris servers, the directory /opt/psf/fontlib (if it exists).

Syntax

USERLIB=*pathlist*

Options and values

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

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

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

Important: The total number of all characters in the string of path names cannot exceed 4095 bytes.

USERMASK

Required

No

Default Value

(None)

Data Type

AFP, Line

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. If the input data is not ASCII, the string must be specified in hexadecimal. For example, when the code page is 500 and the input data is EBCDIC:

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

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 FIELD parameter on page 56.

USERPATH

Specifies the names of user directories that contain TrueType and OpenType fonts or data object resources that are installed with a resource access table (RAT) such as color management resources (CMRs). TrueType and OpenType fonts are Unicode-enabled AFP fonts that are not defined by the Font Object Content Architecture (FOCA).

By convention, resources that are specified with the USERPATH parameter are typically used by one user, as opposed to the system resources that are shared by many users (for example, those specified with the FONTPATH or OBJCPATH parameters).

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

Syntax

USERPATH=*pathlist*

Options and values

The *pathlist* is any valid search path. You must use a colon (:) in AIX and z/OS or a semicolon (;) in Windows to separate multiple paths. For example:

AIX or Windows

```
userpath=/jdoe/fonts/truetype:/jdoe/fonts/truetype/myfonts/
```

z/OS

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

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

Important: The total number of all characters in the string of path names cannot exceed 4095 bytes.

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.

See *IBM Content Manager OnDemand: Messages and Codes* 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.

Chapter 5. User exits and attributes of the input file

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

This section describes the following topics:

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

User programming exits

IBM provides several sample programming exits to assist you in customizing ACIF. On AIX, the sample programs are in /usr/lpp/ars/exits/acif. On HP-UX and Solaris, the sample programs are in /opt/ondemand/exits/acif. On Windows, the sample programs are in \Program Files\IBM\OnDemand for Windows\exits\acif. The compiled programs do not have to be placed into any specific directory, they only have to match the directory specified on the ACIF exit parameter: INPEXIT, OUTEXIT, INDEXEXIT RESEXIT. Each of these parameters is described in Chapter 3, “Parameter reference,” on page 49.

Important: IBM provides compiled versions of the sample user exit programs. If you make changes to the sample user exit programs or create your own user exit programs, you must compile them before the programs can be used by ACIF.

Because the header files for the user exit programs can change between releases and fix packs, IBM recommends that you recompile all user exit programs after upgrading the ACIF component of Content Manager OnDemand. Failure to do so may cause unexpected results when indexing data with ACIF.

See the ACIF README file on the Content Manager OnDemand server CD for more information about supported compilers and files provided by IBM to assist you with programming the user exits.

Use of the programming exits is optional.

IBM provides the following ACIF sample exits:

apkinp.c	Input record exit
apkind.c	Index record exit
apkout.c	Output record exit
apkres.c	Resource exit

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

apka2e.c	Converts ASCII stream data to EBCDIC stream data.
-----------------	---

asciinp.c	Converts unformatted ASCII data that contains carriage returns and form feeds into a record format that contains an American National Standards Institute (ANSI) carriage control character. This exit encodes an ANSI carriage control character in byte 0 (zero) of every record.
asciinpe.c	Converts unformatted ASCII data into a record format as does asciinp.c , and then converts the ASCII stream data to EBCDIC stream data.

The **apkexits.h** C language header file for all ACIF exit programs is also provided.

Input record exit

ACIF provides an exit that enables you to add, delete, or modify records in the input file. You can also use the exit to insert indexing information. For example, imagine that you have unformatted ASCII data, such as a phone bill which contains two consecutive asterisks (**) to distinguish each page break. You can use the exit to add carriage control characters to the data so that ACIF can recognize where page breaks should occur. 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. The exit can request that the record be discarded, processed, or processed and control returned to the exit for the next input record. The largest record that can be processed is 32756 bytes. This exit is not called when ACIF is processing resources from directories (libraries on z/OS).

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 125 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 19 contains a sample C language header that describes the control block that is passed to the exit program.

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

Figure 19. Sample Input Record Exit C Language Header

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

work (Bytes 1–4)

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

pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the input file” on page 113 for more information on the format of this data structure and the information it contains.

record (Bytes 9–12)

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

reserved1 (Bytes 13–16)

These bytes are reserved for future use.

recordln (Bytes 17–18)

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

reserved2 (Bytes 19–20)

These bytes are reserved for future use.

request (Byte 21)

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

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

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

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

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

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

eof (Byte 22)

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

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

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

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

Figure 20 contains a sample DSECT that describes the control block for the z/OS exit program.

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

Figure 20. z/OS Sample Input Record Exit DSECT

The address of the control block containing the following parameters is passed to the input record exit. For z/OS, 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 113 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.

Using the user input record exits

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

To execute the **apka2e** input record exit program, specify these parameters using either the Keyboard Edit function of the administrative client or the graphical indexer.

```
inpexit=apka2e  
cc=yes  
cctype=a
```

Also, make sure that the directory where the **apka2e** input record exit program resides is included in the **INPEXIT** parameter (specify the full path name) or if running ACIF from the command line, in the **PATH** environment variable (or specify the full path name).

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

To execute the **asciinp** input record exit program, specify these parameters using either the Keyboard Edit function of the administrative client or the graphical indexer.

```
inpexit=asciinp
cc=yes
cctype=z
fileformat = stream,(newline=X'0A')
```

Also, make sure that the directory where the **asciinp** input record exit program resides is included in the INPEXIT parameter (specify the full path name) or if running ACIF from the command line, in the **PATH** environment variable (or specify the full path name).

Note: If the indexing parameters were created before **asciinp** processed the file, the following change must be made to the TRIGGER and FIELD parameters: All column offsets must be increased by 1 to account for the fact that the **asciinp** exit inserts an extra byte at the beginning of each record.

The **asciinpe** input record exit program combines both user input record exits described above. If the data contains the ASCII form feed character (X'0C'), the **asciinpe** exit must be used to insert ANSI carriage controls so that the data can be loaded into OnDemand. If you are running ACIF with CONVERT=YES, set cctype=z. If you are running ACIF with CONVERT=NO, set cctype=a. To execute, set the following parameters in your ACIF parameter file:

```
inpexit=asciinpe
cc=yes
cctype=z
cpgid=500
fileformat = stream,(newline=X'0A')
```

Also, make sure that the directory where the **asciinpe** input record exit program resides is included in the INPEXIT parameter (specify the full path name) or if running ACIF from the command line, in the **PATH** environment variable (or specify the full path name).

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

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

For more information on using and modifying these programs, refer to the comments at the beginning of the **asciinp.c** source file, which is provided with OnDemand.

Note: If the indexing parameters were created before **asciinpe** processed the file, the following changes must be made to the TRIGGER, FIELD, and INDEX parameters:

1. All column offsets must be increased by 1 to account for the fact that the **asciinpe** exit inserts an extra byte at the beginning of each record.
2. All TRIGGER values, constant FIELD values, and INDEX names must be encoded in EBCDIC.

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

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

Figure 21. Sample Index Record Exit C Language Header

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

work (Bytes 1–4)

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

pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the input file” on page 113 for more information on the format of this data structure and the information it contains.

record (Bytes 9–12)

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

recordln (Bytes 13–14)

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

request (Byte 15)

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

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

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

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

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

eof (Byte 16)

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

When **eof** is signaled (**eof** value='Y'), the last record has already been presented to the index exit. The record pointer is no longer valid. The following are the only valid values for this parameter:

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

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

Figure 22 contains a sample DSECT that describes the control block that is passed to the z/OS 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 22. z/OS Sample Index Record Exit DSECT

The address of the control block containing the following parameters is passed to the index record exit. For z/OS, 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 113 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, allows the exit program to return control to ACIF. The exit program cannot change this parameter.

Output record exit

Using the output record exit, you can modify or ignore the records ACIF writes into the output document file. The program invoked at this exit is defined by the **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 23 contains a sample C language header that describes the control block passed to the exit program.

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

Figure 23. Sample Output Record Exit C Language Header

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

work (Bytes 1–4)

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

pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the input file” on page 113 for more information on the format of this data structure and the information contained in it.

record (Bytes 9–12)

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

recordln (Bytes 13–14)

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

request (Byte 15)

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

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

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

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

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

eof (Byte 16)

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

When **eof** is signaled (**eof** value='Y'), the last record has already been presented to the output exit. The record pointer is no longer valid. The following are the only valid values for this parameter:

Y Specifies that the last record has been written.

N Specifies that the last record has not been written.

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

Figure 24 contains a sample DSECT that describes the control block passed to the z/OS 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 24. z/OS Sample Output Record Exit DSECT

The address of the control block containing the following parameters is passed to the output record exit. For z/OS, 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 113 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'01'. 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, allows the exit program to return control to ACIF. The exit program cannot change this parameter.

Important: If the output and index file from ACIF are stored in Content Manager OnDemand, do not delete output records using Output record exit as this might invalidate the indexing information.

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 that 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 directory (library in z/OS). 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 25 contains a sample C language header that describes the control block that is passed to the exit program.

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

Figure 25. Sample Resource Exit C Language Header

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

work (Bytes 1–4)

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

pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the input file” on page 113 for more information on the format of this data structure and the information presented.

resname (Bytes 9–16)

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

restype (Byte 17)

Specifies the type of resource the name refers to. This is a 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'FB'** Specifies a page segment.
- X'FC'** Specifies an overlay.

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

- Page definition

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

- Form definition

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

- Coded fonts

If you specify MCF2REF=CF, ACIF includes coded fonts. If MCF2REF=CPCS (the default), ACIF processes coded fonts to determine the names of the code pages and font character sets they reference. This is necessary in creating Map Coded Font-2 (MCF-2) structured fields.

request (Byte 18)

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

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

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

A value of X'00' on entry to the exit program specifies that the resource be processed. If you want to ignore the resource, change the **request** byte value to X'01'. Any value other than X'01' will cause ACIF to process the resource.

Figure 26 contains a sample DSECT that describes the control block that is passed to the z/OS 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 26. z/OS Sample Resource Exit DSECT

The address of the control block containing the following parameters is passed to the resource exit. For z/OS, 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 113 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, allows the exit program to return control to ACIF. The exit program cannot change this parameter.

Non-Zero return codes

If ACIF receives a non-zero return code from any exit program, ACIF issues message 0425-412 and terminates processing. See *IBM Content Manager OnDemand: Messages and Codes* for information about ACIF messages.

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 27 shows the format of the data structure in UNIX and Windows.

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

Figure 27. Sample Print File Attributes C Language Header

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

cc (Bytes 1–3)

The value of the **cc** parameter as specified 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; that is, padding blanks are added to the end of the string. For example, if you specify **PAGEDEF=P1TEST**, the page definition value in the above data structure is P1TEST (the string P1TEST followed by two blank characters).
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.

Figure 28 shows the format of the data structure in z/OS.

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 28. z/OS Sample Print File Attributes DSECT

The address of the control block containing the following parameters is passed to the input record exit. For z/OS, 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 (the string P1TEST followed by two blank characters).
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.

The following topics may provide information that is helpful when using ACIF.

- Working with control statements that contain numbered lines
- Placing TLEs in named groups
- Working with file transfer
- Understanding how ANSI and machine carriage controls are used
- Understanding common methods of transferring files from other systems to OnDemand servers:
 - Physical media such as tape
 - PC file transfer program
 - FTP
 - Download
- Using the Invoke Medium Map (IMM) structured field
- Indexing considerations
- Concatenating the resource file and the document
- Concatenating resources to an AFP file
- Specifying the IMAGEOUT parameter
- Running with inline resources
- Writing inline resources to the output file

Working with control statements that contain numbered lines

Note: This topic contains information relevant to running ACIF on z/OS systems.

You sometimes can receive unexpected results when data set names are continued and the control statements have line numbers in columns 73-80, because ACIF reads all 80 columns of the control statements for processing purposes. (ACIF attempts to use the line number as a data set name, and issues 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.

Placing TLEs in named groups

To avoid having ACIF terminate with errors, 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 contains composed (AFP data stream) pages, page-level TLEs (TLE records after the AEG), and no named groups (BNG/ENG), ACIF 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.

Working with file transfer

ACIF needs to know two things about a file in order to print it:

- The length of each print record
- What kind of carriage control is used

As simple as this sounds, it is the source of most of the difficulty people have printing with ACIF in a workstation environment.

ACIF processes print records. A record is a sequence of contiguous characters, usually representing a printed line or a MO:DCA (AFP data stream) structured field.¹ Each record has a defined boundary or length. Some files contain information in each record that describes the record's length; these are called variable-length files. Other files require an external definition of length; these are called fixed-length files.

- **Variable-length files**

Variable-length files may use a length prefix that provides the length of a record in the file. For variable-length files that contain records with a length prefix, each record in the file contains a two-byte length prefix that provides the length of the record. The length prefix is a 16-bit binary number. The value of the length prefix does not include the two-byte length prefix. Use the **FILEFORMAT=RECORD** control statement to identify a file that contains records with length prefixes.

Variable-length files may use a separator or delimiter to indicate the end of a record, instead of using a length prefix. All of the bytes up to, but not including, the delimiter are considered to be part of the record. In UNIX, the delimiter is X'0A' (In Windows, the delimiter is X'0D0A'). If the file uses EBCDIC encoding, the delimiter is X'25'. Use the **FILEFORMAT=STREAM** control statement to designate files that use delimiters to indicate record boundaries.

ACIF reads the first six bytes and tests for all ASCII characters², to determine if a file is encoded in ASCII or EBCDIC. If no non-ASCII characters are found, ACIF assumes the file uses the ASCII newline character, X'0A'.

Otherwise, ACIF assumes the file uses the EBCDIC newline character, X'25'.

Because an input file can misguide ACIF, either intentionally or by accident, a set of rules has been established to determine how ACIF will interpret how a file will be processed. Table 2 lists the possible combinations.

Table 2. Data type and Newline character combinations

Data Type	Newline Character
All EBCDIC	EBCDIC X'25'
All EBCDIC	ASCII X'0A'
All ASCII	EBCDIC X'25'
All ASCII	ASCII X'0A'

1. Structured fields are similar to print commands.

2. Code points from X'00' to X'7F'

Note: These combinations are possible only if a file contains a prefix with a string that indicates a different code set than actually exists. For EBCDIC data with ASCII newlines, use X'0320202020200A'. For ASCII data with EBCDIC newlines, use X'03C1C1C1C1C125'.

- **Fixed-length files**

Fixed-length files contain records that are all the same length. No other separators or prefixes or self-identifying information exists that indicates the record length. You must know the record length and use the **FILEFORMAT=RECORD,nnn** control statement, where *nnn* represents the length of each record.

For variable- and fixed-length files using length prefixes, MO:DCA structured fields are treated as a special case. All such structured fields are self-identifying and contain their own length. They need not contain a length prefix to be correctly interpreted, but will be processed correctly if there is a length prefix.

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. Table 3 lists the standard ANSI characters.

Table 3. Standard ANSI characters

ANSI	Command
space	Single space the line and print
0	Double space the line and print
-	Triple space the line and print
+	Do not space the line and print
1	Skip to channel 1 (the top of the form, by convention)
2 through 9	Skip to hardware-defined position on the page
A, B, or 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. Table 4 on page 120 lists the typical machine controls.

Table 4. Typical machine controls

Machine	Command
X'09'	Print the line and single space
X'11'	Print the line and double space
X'19'	Print the line and triple space
X'01'	Print the line and don't space
X'0B'	Space one line immediately (don't print)
X'89'	Print the line, then skip to channel 1 (top of form, by convention)
X'8B'	Skip to channel 1 immediately (don't print)

Note that machine controls print before performing any required spacing. There are many more machine control commands than ANSI. Carriage controls may be present in a print file or not, but every record in the file must contain a carriage control if the controls are to be used. If the file contains carriage controls, but **CC=NO** is specified to ACIF, the carriage controls will be treated as printing characters. If no carriage controls are specified, the file will be printed as though it were single spaced.

Common methods of transferring files

You can transfer files from other systems to OnDemand servers using a variety of methods. Each method results in a different set of possible outputs. Some methods produce output that cannot be used by ACIF. Methods commonly used to transfer files from other systems to OnDemand servers and produce output that ACIF can use are:

- Physical media (such as tape)
- PC file transfer program
- FTP
- Download

Physical media

Normally, you can copy fixed-length files without any transformation using a physical media, such as tape.

PC file transfer program

You may transfer files from other systems to OnDemand servers by using an implementation of the most common PC file transfer program (IND\$FILE). You may also transfer files from a host to a personal computer. The variety of possible parameters that can affect printing are host-dependent. IBM recommends the following:

- For z/OS, 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 control statement **INPEXIT=asciipe** to remove the otherwise unprintable carriage return (X'0D') that is inserted in the file.
- For VSE files, additional file transfer parameters are available.

- For files with machine carriage control, you can specify BINARY, CRLF and CC. This provides an EBCDIC file with correct carriage controls separated by ASCII newlines and carriage returns.

FTP

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

Download

You can use Download to transmit a print data set from the JES spool to file systems on OnDemand servers. 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 an OnDemand server. You can use other JCL parameters, such as FORM and DATASET to determine the application group and application to load. Download transmits data in binary format.

To conserve space and increase transmission speed, Download truncates a record if it contains one or more blank characters (X'40') at the end of the record. As a result, after transmitting a report to the server, some records may contain fewer characters than the assumed record length. If the location of a FIELD begins outside the actual length of a record, ACIF fails unless you specify a DEFAULT value. For example, a report on the 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. During transmission of the file, Download eliminates the blank characters from the end of all records that contain X'40' in columns 129 through 133. To prevent ACIF from failing, you must define a DEFAULT value for the field. For example:

```
FIELD2=1,129,4,(DEFAULT=X'D5D6D5C5')
```

In the example, if a record is not 129 bytes in length, ACIF generates the value NONE (X'D5D6D5C5') for FIELD2.

Other considerations for transferring files

Conventional file transfer programs cannot correctly handle the combination of variable-length files, which contain bytes that cannot be translated from their original representation to ASCII, and may also contain machine control characters, mixed line data and structured fields, or special code points that have no standard mapping.³ 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.)

3. When ASCII is specified, for example, the file transfer program may destroy the data in translation. When binary is specified, the file transfer program may not be able to indicate record lengths.

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

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

Using the Invoke Medium Map (IMM) structured field

Retrieval programs must be able to detect which medium map is active, to ensure that pages are reprinted (or viewed) using the correct medium map. To ensure that the correct medium map is used, use the Active Medium Map triplet and the Medium Map Page Number triplet (from the appropriate Index Element [IEL] structured field in the index object file), which designate the name of the last explicitly invoked IMM structured field and the number of pages produced since the IMM was invoked. The retrieval system can use this information to dynamically create IMM structured fields at the appropriate locations when it retrieves a group of pages from the archived document file.

Indexing considerations

The index object file contains Index Element (IEL) structured fields that identify the location of the tagged groups in the print file. The tags are contained in the Tagged Logical Element (TLE) structured fields.

The structured field offset and byte offset values are accurate at the time ACIF creates the output document file. However, if you extract various pages or page groups for viewing or printing, you will have to dynamically create from the original a temporary index object file that contains the correct offset information for the new file. For example, assume the following:

- ACIF processed all the bank statements for 6 branches, using the account number, statement date, and branch number.
- The resultant output files were archived using a system that allowed these statements to be retrieved based on any combination of these three indexing values.

If you wanted to view all the bank statements from branch 1, your retrieval system would have to extract all the statements from the print file ACIF created (possibly using the IELs and TLEs in the index object file) and create another document for viewing. This new document would need its own index object file containing the correct offset information. The retrieval system would have to be able to do this.

Under some circumstances, the indexing that ACIF produces may not be what you expect, for example:

- If your page definition produces multiple-up output, and if the data values you are using for your indexing attributes appear on more than one of the multiple-up subpages, ACIF may produce two indexing tags for the same physical page of output. In this situation, only the first index attribute name will appear as a group name, when you are using OnDemand. To avoid this, specify a page definition that formats your data without multiple-up when you run 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.

Concatenating resources to an AFP file

A resource group can be created and stored in a file by using the ARSACIF 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 lists the parameters used to create a resource file using the ARSACIF program. The parameters process an AFP file named `credit.afp`, which is an AFP file that contains no indexing information or inline resources. The example is for an AIX system. 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 `parms.acif`:

```
CC=YES
CCTYPE=A
RESTYPE=OVLY,PSEG,FDEF
INPUTDD=credit.afp
OUTPUTDD=/dev/null
INDEXDD=/dev/null
RESOBJDD=credit.res
USERLIB=/usr/resources
```

Command used to generate the resource group file:

```
arsacif parmd=parms.acif
```

Command to concatenate the resource group file and the AFP file:

```
cat credit.res credit.afp > credit.out
```

You can then process the `credit.out` file with the indexing program if you want to index the data.

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.

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 **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.

Here is the recommended order that the resources should appear in the input file:

```
FORMDEF
CHARACTER SETS
CODE PAGES
PAGE SEGMENTS
OVERLAYS
PAGEDEF
```

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.

Chapter 7. ACIF data stream information

General-use Programming Interface and Associated Guidance Information is contained in this section.

This section contains the following topics:

- Tag Logical Element (TLE) structured field
- Formats of the resource file
- Understanding how ACIF processes fully composed AFP files

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 an index name and an index value. For the purposes of illustration, each field within the structured field is listed on a separate line. X' ' denotes hexadecimal data, and " " denotes EBCDIC or character data.

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

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

pages, and you may want to associate specific indexing information at the statement level (for example, account number, date, customer name, and so on). You may also want to index (tag) a specific page within the statement, such as the summary page. The following is an example of a print file that contains TLEs at the group level as well as at the page level:

```
BDT
  BNG
    TLE Account #, 101030
    TLE Customer Name, Mike Smith
    BPG
      Page 1 data
    EPG
    BPG
      Page 2 data
    EPG
    ...
    ...
    BPG
      TLE Summary Page, n
      Page n data
    EPG
  ENG
  ...
EDT
```

ACIF can accept input files that contain both group-level and page-level indexing tags. You can also use the input record exit of ACIF to insert TLE structured fields into an AFP data stream (MO:DCA-P) file, where applicable. The indexing information in the TLE structured field applies to the page or group containing them. In the case of groups, the TLE structured field can appear anywhere between a Begin Named Group (BNG) structured field and the first page (BPG structured field) in the group. In the case of composed-text pages, the TLE structured field can appear anywhere following the Active Environment Group, between the End Active Environment (EAG) and End Page (EPG) structured fields. Although ACIF does not limit the number of TLE structured fields that can be placed in a group or page, you should consider the performance and storage ramifications of the number included.

ACIF does not require the print file to be indexed in a uniform manner; that is, every page containing TLE structured fields does not have to have the same number of tags as other pages or the same type of index attributes or tag values. This allows a great deal of flexibility for the application. When ACIF completes processing a print file that contains TLE structured fields, the resultant indexing information file may contain records of variable length.

Format of the resource file

ACIF retrieves referenced AFP resources from specified directories (libraries in z/OS) and creates a single file (dataset in z/OS) 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 path and file name of the resource group. 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 **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 one-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.

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.

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 enable applications such as archival and retrieval applications to selectively determine the location of a page group or page within the AFP data stream print file, based on its index (tag) values.

The following example shows the general internal format of this object:

```
BDI
  IEL GroupName=G1
    TLE (INDEX1)
    ...
    TLE (INDEXn)
      IEL PageName=G1P1
        TLE (INDEX1)
        ...
        TLE (INDEXn)
      ...
      IEL PageName=G1Pn
    ...
  IEL GroupName=Gn
    TLE (INDEX1)
    ...
    TLE (INDEXn)
      IEL PageName=GnP1
        TLE (INDEX1)
        ...
        TLE (INDEXn)
      ...
      IEL PageName=GnPn
EDI
```

The example illustrates an index object file containing both page-level and group-level Index Element (IEL) and Tag Logical Element (TLE) structured fields.

Group-level Index Element (IEL) structured field

If **INDEXOBJ=GROUP** is specified, ACIF creates an index object file with the following format:

```
BDI
  IEL Groupname=G1
    TLE
    ...
    TLE
  ...
  IEL Groupname=Gn
    TLE
    ...
    TLE
EDI
```

This format is useful to reduce the size of the index object file, but it allows manipulation only at the group level; that is, you cannot obtain the offset and size

information for individual pages. You also lose any indexing information (TLEs) for pages; the TLE structured fields for the pages still exist in the output print file, however.

Page-level Index Element (IEL) structured field

If **INDEXOBJ=ALL** is specified, ACIF creates an index object file with the following format:

```
BDI
  IEL Groupname=G1
    TLE
    ...
    IEL Pagename=G1P1
      TLE
      ...
    ...
    IEL Pagename=G1Pn....
  ...
  IEL Groupname=Gn
    TLE
    ...
    IEL Pagename=GnP1
      ...
    IEL Pagename=GnPn
      TLE
    ...
EDI
```

This example contains IEL structured fields for both pages and groups. Notice that TLE structured fields are associated with both pages and groups. In this example, where an application created an indexed AFP print file containing both page-level and group-level TLE structured fields, ACIF can create IEL structured fields for the appropriate TLE structured fields.

An index object file containing both page-level and group-level IEL structured fields can provide added flexibility and capability to applications that operate on the files created by ACIF. This type of index object file provides the best performance when you are viewing a file using OnDemand.

Begin Document Index (BDI) structured field

ACIF assigns a null token name (X'FFFF') and an FQN type X'01' triplet to this structured field. The FQN type X'01' value is the file name identified by the **INDEXDD** parameter. ACIF also creates an FQN type X'83' triplet containing the name of the AFP output print file, identified by 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 Chapter 3, "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 129; 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 29. Example of Code Containing Group-Level Indexing

Figure 29 illustrates the one of the formats that 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 30. Example of Code Containing Group- and Page-Level Indexing

Figure 30 illustrates the other format that ACIF can produce when it converts and indexes a print file, generating indexes at both the group and page level.

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 **OUTPUTDD** parameter. 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 **OUTPUTDD** parameter.

ACIF also creates a Coded Graphic Character Set Global Identifier triplet X'01' using the code page identifier specified in the **CPGID** parameter. For more information on the **CPGID** parameter, see the *arsacif* command reference, beginning on page Chapter 3, "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 129, 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

This chapter provides information about using ACIF in the z/OS environment. You can run ACIF on a z/OS system on which the ACIF programs are installed. To run ACIF on a z/OS system requires:

- OS/390 Version 2 Release 10 or later or z/OS Version 1 Release 1 or later
- PSF for OS/390 Version 3 Release 3 or later
- OnDemand version of ACIF, which can be ordered without charge by customers who are entitled to OnDemand. See the README file provided with the OnDemand product package for ordering information.

Sample JCL

Figure 31 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 31. Sample z/OS JCL to Invoke ACIF

About the JCL statements

The JCL statements in Figure 31 are explained as follows. For more information about programming JCL, refer to the *PSF Application Programming Guide*.

USERAPPL

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

PRINTOUT

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

ACIF

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

PARMDD

Defines the DDname for the data set containing the ACIF processing

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

MSGDD

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

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

INPUT

This DD statement defines the print data set to be processed by ACIF. In the sample in Figure 31 on page 139, 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 31 on page 139 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.

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 32 on page 142 shows an example of z/OS 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 32. Example of an z/OS ACIF Application

z/OS libraries

The example ACIF parameters defined the libraries listed in Table 5.

Table 5. Libraries defined in example ACIF parameters

Library Name	z/OS Name
FDEFLIB Form definition library	SYS1.FDEFLIB
FONTLIB Font libraries	SYS1.FONTLIBB SYS1.FONTLIBB.EXTRA
OVLYLIB Overlay library	SYS1.OVERLIB
PDEFLIB Page definition library	SYS1.PDEFLIB
PSEGLIB Page segment library	SYS1.PSEGLIB

ACIF output

The example ACIF job created the output files listed in Table 6.

Table 6. Output files created by example ACIF job

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 OS/390 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. You can then transfer the concatenated file to an OnDemand server and process it with the OnDemand data loading programs.

Figure 33 on page 144 shows z/OS 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 33. Example of z/OS 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.

The resource file must have been created by specifying **RESFILE=SEQ**.

Part 2. Generic indexer reference

This part provides information about the Content Manager OnDemand Generic indexer. You can use the Generic indexer to specify index data for any type of input file that you want to store in the system, although you typically use a format-specific indexer, if one is available. OnDemand includes the following format-specific indexers:

- For Advanced Function Printing (AFP) data and line data, use AFP Conversion and Indexing Facility (ACIF).
- For PDF data, use the IBM Content Manager OnDemand PDF Indexer for Multiplatforms.
- For AFP data and Metacode/DJDE data, use the Xenos transforms.

Reports produced using Crystal Report are a good example of input files that you might index with the Generic indexer, since these reports are saved in a proprietary format.

Chapter 11. Overview

OnDemand provides the Generic indexer to allow you to specify indexing information for input data that you cannot or do not want to index with ACIF, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms, or Xenos. For example, suppose that you want to load files into the system that were created with a word processor. The files can be stored in OnDemand in the same format in which they were created. The files can be retrieved from OnDemand and viewed with the word processor. However, because the files do not contain AFP data, line data, PDF data, or Metacode/DJDE data, you cannot index them with the other indexers that are supported by OnDemand. You can specify index information about the files in a format that is used by the Generic indexer and load the index data and files into the system. Users can then search for and retrieve the files by using one of the OnDemand client programs.

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

Loading data

Most customers use the ARSLOAD program to load data into 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 file). For example, the ARSLOAD program can invoke the Generic indexer to process the parameter file and generate the index data. The ARSLOAD program can then add the index information to the database and load the input files or documents specified in the parameter file on to storage volumes.

There are two ways to run the ARSLOAD program:

- **Daemon mode.** The ARSLOAD program runs as a daemon (UNIX servers) or service (Windows servers) to periodically check a specified directory for input files to process. When running the ARSLOAD program in daemon mode, a dummy file with the file type extension of .ARD is required to initiate a load process. In addition, the Generic indexer parameter file (.IND) must be located in the specified directory. The GROUP_FILENAME: parameter in the .IND file specifies the full path name of the actual input file to be processed.
- **Manual mode.** The ARSLOAD program is run from the command line to process a specific file. When running the ARSLOAD program in manual mode, specify only the *name* of the file to process. The ARSLOAD program adds the .IND file

name extension to the name that you specify. For example, if you specify `arsload ... po3510`, where `po3510` is the name of the input file, the ARSLOAD program processes the `po3510.ind` Generic indexer parameter file. The `GROUP_FILENAME:` parameter in the Generic indexer parameter file specifies the full path name of the actual input file to be processed.

After successfully loading the data, the system deletes the input file that is specified on the `GROUP_FILENAME:` parameter if the file name extension is `.OUT`, and for daemon mode processing, the rest of the input file name is the same as the `.ARD` file name. The system also deletes the `.IND` file (the Generic indexer parameter file) and the `.ARD` file (the dummy file that is used to initiate a load process when the ARSLOAD program is running in daemon mode).

The following shows an example of file names in daemon processing mode:

```
MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMSST.ARD
MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMSST.ARD.IND
MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMSST.ARD.OUT
```

The `MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMSST.ARD` file is the dummy file that triggers a load process in daemon mode. The `MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMSST.ARD.IND` file is the Generic indexer parameter file, and contains a `GROUP_FILENAME:` parameter that specifies the input file to process: `MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMSST.ARD.OUT`. After successfully loading the data, the system deletes all three files.

Processing AFP data

You can specify a parameter file for input files that contain AFP resources and documents and process them with the Generic indexer. However, when you specify the parameter file:

- The starting location (byte offset) of the first AFP document in the input file 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 file. The actual starting location of the first document in the input file 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 file should be calculated using the length of and offset from the previous document in the input file. For example:

Table 7. How the starting locations of the other documents in the input file should be calculated

AFP structured field	Physical file offset/length	Generic index file GROUP_OFFSET/ GROUP_LENGTH
Begin Resource Group/End Resource Group	0 / 282	
Begin Document 1/End Document 1	282 / 6223	0 / 6223
Begin Document 2/End Document 2	6505 / 6267	6223 / 6267
Begin Document 3/End Document 3	12772 / 6588	12490 / 6588

Table 7. How the starting locations of the other documents in the input file should be calculated (continued)

AFP structured field	Physical file offset/length	Generic index file GROUP_OFFSET/ GROUP_LENGTH
Begin Document 4/End Document 4	19360 / 5876	19078 / 5876
Begin Document 5/End Document 5	25236 / 5895	24954 / 5895
Begin Document 6/End Document 6	31131 / 5943	30849 / 5943

The Generic indexer determines where the AFP resources end in the file 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 one or more input files that you want to load into the system and a parameter file that contains the indexing information for the input files. To use the Generic indexer, you must create a parameter file that contains the indexing information for the input files. This section describes the parameter file that is used by the Generic indexer.

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

- **Comments.** You can place a comment line anywhere in the parameter file.
- **Code page.** You must specify a code page line at the beginning of the parameter file, before you define any 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 file, the number of bytes (characters) that make up the document, and the name of the input file that contains the document.

Important:

1. The parameter names in the parameter file 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:954` is specified, set the locale environment variable to `ja_JP` or some other locale that correctly identifies upper and lower case characters in code page 954.

CODEPAGE:

Specifies the code page of the input data. You must specify one and only one code page. The **CODEPAGE:** line must appear before you specify 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:954` is specified, set the locale environment variable to `ja_JP` or some other locale that correctly identifies upper and lower case characters in code page 954.

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 819 for the input data:

```
CODEPAGE:819
```

COMMENT:

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

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 file must contain one **GROUP_FIELD_NAME:** line for each application group field. (The application group is where you store a file 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 for the application.

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 file 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 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 file 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 file 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 examples of index values:

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

GROUP_FILENAME:

The file name or full path name of the input file. If you do not specify a path, the Generic indexer searches the current directory for the specified file; however, you should always specify the full path name of the input file. On UNIX servers, file and path names are case sensitive.

Each group that you specify in the parameter file 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_FILENAME:studentID
```

```
GROUP_FIELD_VALUE:0012345678
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:/tmp/statements.out
```

If the **GROUP_FILENAME:** line does not contain a value (blank), 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 file 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:/tmp/statements.out
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 file is blank, you must specify the name of the input file when you run the ARSLOAD command.

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

After successfully loading the data, the system deletes the input file that is specified on the **GROUP_FILENAME:** parameter if the file name extension is .OUT, and for daemon mode processing, the rest of the input file name is the same as the .ARD file name. The system also deletes the .IND file (the Generic indexer parameter file) and the .ARD file (the dummy file that is used to initiate a load process when the ARSLOAD program is running in daemon mode). See “Loading data” on page 147 for more information.

Syntax

```
GROUP_FILENAME:fileName
```

Options and values

The character string **GROUP_FILENAME:** identifies the line as containing the input file to process. The string *fileName* specifies the full path name of the input file. You should always specify the full path name of the input file to process. For example:

```
GROUP_FILENAME:/tmp/ondemand/inputfiles/f1b0a1600.out
```

Example

The following are valid file name lines:


```
GROUP_FILENAME:/tmp/statements
GROUP_FILENAME:D:\ARSTMP\statements
GROUP_FILENAME:/tmp/ondemand/inputfiles/f1b0a1600.out
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 file. Each group that you specify in the parameter file 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 file.

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 file of the data to be indexed. Specify 0 (zero) for the first byte (the beginning) of the file. (If processing AFP documents and resources with the Generic indexer, see “Processing AFP data” on page 148.) Each group that you specify in the parameter file 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 file.

Example

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

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

Chapter 13. Parameter file 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 file.

```
COMMENT:
COMMENT: Generic Indexer Example 1
COMMENT: Different input file for each document
COMMENT:
COMMENT: Specify code page of the index data
CODEPAGE:819
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:/arstmp/statement7.out
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:/arstmp/statement8.out
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:/arstmp/statement9.out
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 file.

```
COMMENT:
COMMENT: Generic Indexer Example 2
COMMENT: One input file contains all documents
COMMENT:
COMMENT: Specify code page of the index data
CODEPAGE:819
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:/arstmp/accounting.student information.loan.out
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. IBM Content Manager OnDemand PDF Indexer for Multiplatforms reference

This part provides information about the Content Manager OnDemand IBM Content Manager OnDemand PDF Indexer for Multiplatforms. You can use the IBM Content Manager OnDemand PDF Indexer for Multiplatforms to extract index data from and generate index data about Adobe PDF files that you want to store in Content Manager OnDemand.

Chapter 14. Overview

What is the IBM Content Manager OnDemand PDF Indexer for Multiplatforms?

The Content Manager OnDemand IBM Content Manager OnDemand PDF Indexer for Multiplatforms is a program that you can use to extract index data from and generate index data about Adobe PDF input files. The index data can enhance your ability to store, retrieve, and view documents with Content Manager OnDemand. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms supports PDF Version 1.2 or later 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 and store PDF documents on the server using standard OnDemand functions. To process a document, you must define an Content Manager OnDemand application and application group. As part of the application, you must define the indexing parameters used by the PDF indexer to process input files. You can automate the indexing and loading of data by configuring and running the ARSLOAD program as a daemon (UNIX servers) or service (Windows servers).

After you use the IBM Content Manager OnDemand PDF Indexer for Multiplatforms and the ARSLOAD program to index and store input files in Content Manager OnDemand, you can do the following:

- Use one of the Content Manager OnDemand client programs to view the PDF document or documents created during the indexing and loading process. You can also print pages of the PDF document you are viewing from the Content Manager OnDemand client program. The client programs use Adobe Acrobat to view PDF documents.
- Use the Content Manager OnDemand server print function to print the PDF document. The server print function requires Infoprint.

Figure 34 on page 162 illustrates the process of indexing and loading PDF input files.

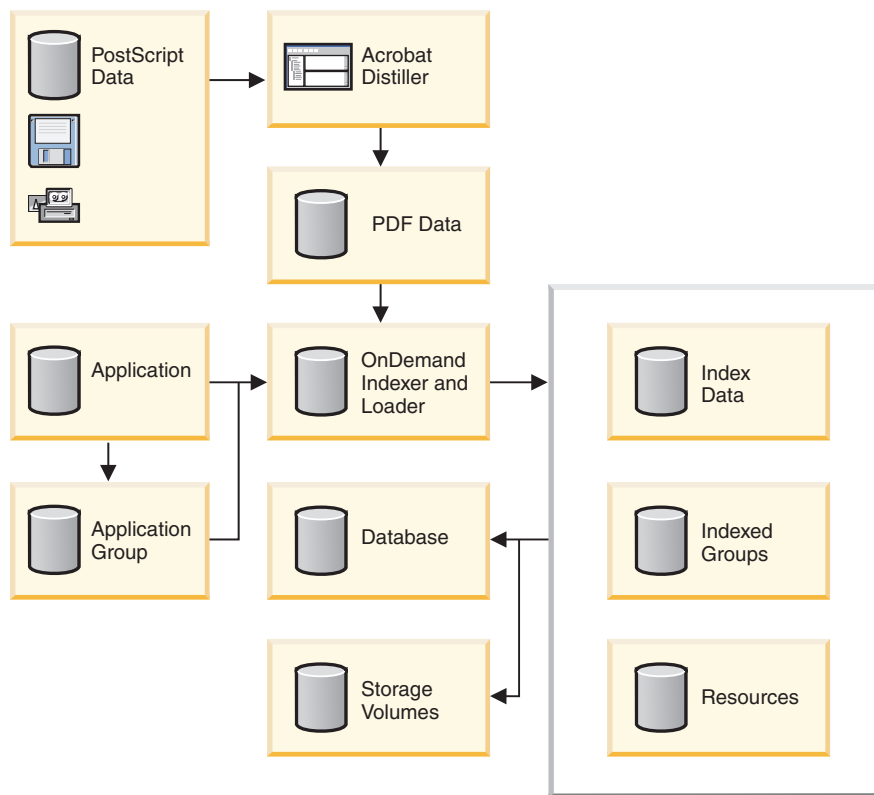


Figure 34. Processing PDF input files in Content Manager OnDemand

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms processes PDF input files. A PDF file is a distilled version of a PostScript file, adding structure and efficiency. A PDF file can be created by Acrobat Distiller or a special printer driver program called a PDFWriter. You can automate the distilling process by configuring and running the Distiller daemon (UNIX servers) or Acrobat Distiller (Windows servers). See the online 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 files, 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 input file and the application Indexer Information tab specifies PDF as the indexer, it automatically calls the IBM Content Manager OnDemand PDF Indexer for Multiplatforms to process the input file. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms processes the PDF input file with indexing parameters that determine the location and attributes of the index data. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms extracts index data from the PDF file and generates an index file and an output file. The output file contains groups of indexed pages. A group of indexed pages can represent the entire input file or, more typically, one or more pages from the input file. If the input file contains logical groups of pages, such as statements or policies, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms can create an indexed group for each statement or policy in the

input file. That way, users can retrieve a specific statement or set of statements, rather than the entire file. The PDF indexer can optionally extract embedded resources from the PDF input files and store them in a resource file. The resource file is loaded into OnDemand at the same time as the output file. After Content Manager OnDemand indexes the data, it stores the index data in the database and the indexed groups and resources on storage volumes. You can automate the data indexing and loading process by configuring and running the ARSLOAD program to run as a daemon (UNIX servers) or service (Windows servers).

How OnDemand uses index information

Every item stored in Content Manager 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 file into Content Manager OnDemand, the ARSLOAD program invokes the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 35 illustrates the index creation and data loading process.

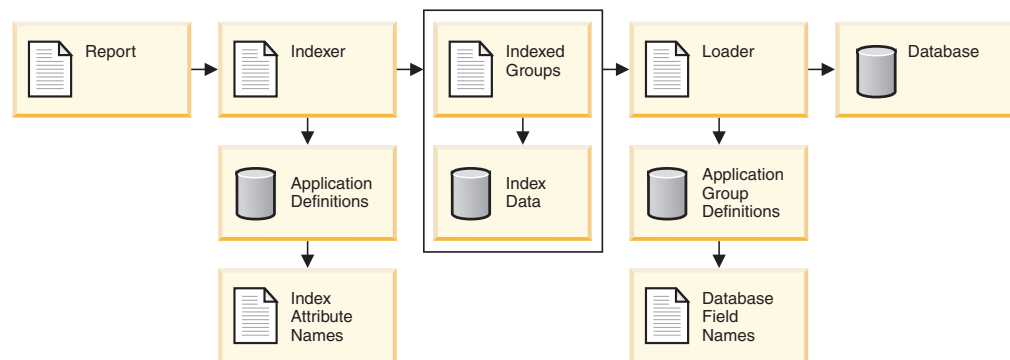


Figure 35. Indexing and loading data

You typically create an application for each report that you plan to store in Content Manager 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 Content Manager 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. Content Manager 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms to identify key items in the print data stream, *tag* these items, and create *index elements* pointing to the tagged items. Content Manager OnDemand uses the tag and index data for efficient, structured search and retrieval. You specify the index information that allows the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms where to locate the attribute value on a page. For example, a tag used to collect customer name index values provides the IBM Content Manager OnDemand PDF Indexer for Multiplatforms with the starting and ending position on the page where the customer name index values appear. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms generates index data and stores it in a generic index file. See Part 2, “Generic indexer reference,” on page 145 for more information about the generic index file.

Coordinate system

The location of the text strings the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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'
```

IBM provides the ARSPDUMP command to help you identify the locations of text strings on the page.

Indexing parameters

Processing parameters can contain index and conversion parameters, options, and values. For most reports, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms requires at least three indexing parameters to generate index data:

- **TRIGGER**

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms uses triggers to determine where to locate data. A trigger instructs the IBM Content Manager OnDemand PDF Indexer for Multiplatforms to look for certain information in a specific location on a page. When the IBM Content Manager OnDemand PDF Indexer for Multiplatforms finds the text string in the input file that contains the information specified in the trigger, it can begin to look for index information.

- The IBM Content Manager OnDemand PDF Indexer for Multiplatforms compares words in the input file with the text string specified in a trigger.
- The location of the trigger string value must be identified using the x,y coordinate system and page offsets.
- A maximum of 16 triggers can be specified.
- All triggers must match before the IBM Content Manager OnDemand PDF Indexer for Multiplatforms can begin to locate index information.

- **FIELD**

The field parameter specifies the location of the data that the IBM Content Manager OnDemand PDF Indexer for Multiplatforms uses to create index values.

- Field definitions are based on TRIGGER1 by default, but can be based on any of 16 TRIGGER parameters.
- The location of the field must be identified using the x,y coordinate system and page offsets.
- A maximum of 32 fields can be defined.
- A field parameter can also specify all or part of the actual index value stored in the database.

- **INDEX**

The index parameter is where you specify the attribute name and identify the field or fields on which the index is based. IBM recommends that you name the attribute the same as the application group database field name.

- The IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms creates a new group and extracts new index values when one or more of the index values change.

Figure 36 on page 166 depicts a portion of a page from a sample input file. The text strings that determine the beginning of a group and the index values are enclosed in rectangles.

3.25													
0.75				0.25		1.00				0.75		0.50	
Page 001													

- 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

Indexing with Metadata Indexes

An Adobe PDF document can contain metadata, which is general information such as title and author that applies to the entire document. You typically create the document's metadata when the document is created and can modify the metadata at any time. For more information on metadata, see the *Adobe PDF Reference, 2nd edition*, ISBN 0-201-61588-6.

When INDEXMODE=METADATA is specified, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms extracts fields from the Document Information Dictionary that correspond to the following metadata keywords, if they exist, and place their values into the .ind file:

- Title
- Author
- Subject
- Keywords
- Creator
- Producer
- CreationDate
- ModDate
- Trapped

The metadata keywords are the group field names within the .ind file and can be mapped to the application group fields in the application. You can opt not to map any group field names. Because the metadata keywords apply to the entire document, you can index the document only as one group. If TRIGGER, FIELD, or INDEX parameters are specified, they are ignored. Metadata indexing cannot be combined with indexing using a TRIGGER. If the document contains none of these metadata fields, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms issues the following error message and stops processing:

```
ARS4940 Index not found by page page number
```

where *page number* is the number specified in the INDEXSTARTBY parameter.

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms is unaware of the date format of the following metadata fields:

```
CreationDate
ModDate
```

As a result, date formatting takes place during post-processing.

The only parameter required for metadata indexing is:

```
indexmode=metadata
```

Here is an example of an index file created by Metadata indexing:

```
COMMENT:
COMMENT: Generic Indexer Format
COMMENT:
COMMENT:
COMMENT: Code Page of the Index Data
CODEPAGE:5348
COMMENT: Index Field(s)
GROUP_FIELD_NAME:Title
GROUP_FIELD_VALUE:Content Manager OnDemand for Multiplatforms: Administrator's Guide
GROUP_FIELD_NAME:Author
GROUP_FIELD_VALUE:IBM
GROUP_FIELD_NAME:CreationDate
GROUP_FIELD_VALUE:20001010160019
COMMENT: Index Offsets and Length
GROUP_OFFSET:0
GROUP_LENGTH:759720
GROUP_PAGES:387
GROUP_FILENAME:\pdfoutput\admin.pdf
COMMENT:
COMMENT:
COMMENT:
COMMENT: End Generic Indexing File
```

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 the IBM Content Manager OnDemand PDF Indexer for Multiplatforms uses to identify groups and locate index data. Then, create the indexing parameters using the administrative client.

Content Manager OnDemand provides the ARSPDUMP command to help you determine the location of trigger and field string values in the input data. The ARSPDUMP command processes one or more pages of sample report data and generates an output file. The output file contains one record for each text string on a page. Each record contains the x,y coordinates for a box imposed over the text string (upper left, lower right).

The process works as follows:

- Obtain a printed copy of the sample report.
- Identify the string values that you want to use to locate triggers and fields
- Identify the number of the page where each string value appears. The number is the *sheet number*, not the page identifier. The sheet number is the order of the page as it appears in the file, beginning with the number 1 (one), for the first page in the file. A page identifier is user-defined information that identifies each page (for example, iv, 5, and 17-3).
- Process one or more pages of the report with the ARSPDUMP command.
- In the output file, locate the records that contain the string values and make a note of the x,y coordinates.
- Create TRIGGER and FIELD parameters using the x,y coordinates, page number, and string value.

Indexing parameters are part of the Content Manager OnDemand application. The administrative client provides an edit window 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 a PDF input file.

Important: If you plan to use the report wizard or the graphical indexer to process PDF input files, then you must first install Adobe Acrobat on the PC from which you plan to run the administrative client. Content Manager 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\Content Manager 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 Indexer Information tab of the application. After you open an input file in the graphical indexer, you define triggers, fields, and indexes. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 Content Manager 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms attempts to locate the specified string in the specified location. When a match occurs, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 button 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 at any time for assistance with using the graphical indexer.
8. 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 mouse button one. 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.
 - Click the Define a Trigger button on the toolbar to open 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.
 - 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, select the trigger. The graphical indexer highlights the text string in the current document. Select 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.
9. 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 mouse button one. 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.

- Click the Define a Field button 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 Content Manager 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, click Field 1. The graphical indexer highlights the text string in the current document. Select 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.
10. Click the Create Indexer Parameters and Fields Summary toolbar button. Use the Create Indexer Parameters and Fields Summary dialog box to create and view a summary of the indexing parameters and field values. See theChapter 16, “Parameter reference,” on page 177 for details about the indexing parameters.
 11. When you have finished defining all of the triggers, fields, and indexes, close the report window.
 12. Click Yes to save the changes to the indexer parameters.
 13. On the Sample Data window, click Next to continue with the report wizard.

PDF Resource Collection

The PDF reports that you store in Content Manager OnDemand might contain embedded resources such as fonts and images. When the report is indexed, it is usually broken up into smaller pieces, and the resources are placed into each new report. Because each new report contains its own resources, the size of the indexed reports can become much larger than the original PDF reports.

In order to decrease the size of the indexed reports, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms can optionally extract these resources from the PDF reports and place them in a resource file. Content Manager OnDemand loads the resource file at the same time as it loads the indexed report files. When a report is retrieved for viewing or printing, the resources are reinserted into the report, and then the report is sent to the client.

A PDF report might contain no resources if it uses only the fourteen standard fonts that are listed in the PDF reference. These fonts are guaranteed to be available on the client, therefore, they are not embedded in the report.

The resources that the IBM Content Manager OnDemand PDF Indexer for Multiplatforms collects are based on the value of the RESTYPE parameter. The following table lists values for this parameter. For a complete list, see “RESTYPE” on page 189.

Table 8. Available values for the RESTYPE parameter

RESTYPE	Meaning	Why
NONE	Do not collect resources.	Report does not contain resources, or the resources are small.
ALL	Collect fonts and images.	To save space that is used to store the reports.
FONT	Collect fonts only.	To save space that is used to store the reports. Report contains fonts only.
IMAGE	Collect images only.	To save space that is used to store the reports. Report contains images only.
FONT, IMAGE	Collect fonts and images.	To save space that is used to store the reports.

There is no resource exit for the IBM Content Manager OnDemand PDF Indexer for Multiplatforms.

Chapter 15. PDF indexing system requirements

Adobe software requirements

For Adobe software requirements, see:

<http://www.ibm.com/support/docview.wss?rs=129&uid=swg27012104>

Specifying the location of Adobe fonts

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms must be able to access fonts to insert appropriate information in a PDF output file. If a font is referenced in an input file but is not available on the system, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms will substitute a font, usually Courier.

All installations should verify that the standard Adobe font files are installed in the standard font directory. For installations that plan to use the IBM Content Manager OnDemand PDF Indexer for Multiplatforms to access DBCS fonts, verify the locations of the DBCS font files and export or add the ACRO_RES_DIR and PSRESOURCEPATH environment variables.

The standard Adobe fonts should always be installed in the standard font directory on the system. Table 9 shows the default locations.

Table 9. Location of Standard Adobe Font Files

Platform	Standard Font Directory
AIX	/usr/lpp/Acrobat3/Fonts
HP-UX	/opt/Acrobat3/Fonts
Solaris	/opt/Acrobat3/Fonts
Windows	\Windows\Fonts

DBCS font files must be installed in the directories specified in Table 10 so that the IBM Content Manager OnDemand PDF Indexer for Multiplatforms can access DBCS fonts that may be required to process a PDF input file.

Table 10. Location of Adobe DBCS Font Files

Platform	Font Directory	CMap File Directory	CIDFont File Directory
AIX	/usr/lpp/Acrobat3/Fonts	/usr/lpp/Acrobat3/Fonts/Resource/CMap	/usr/lpp/Acrobat3/Fonts/Resource/CIDFont
HP-UX	/opt/Acrobat3/Fonts	/opt/Acrobat3/Fonts/Resource/CMap	/opt/Acrobat3/Fonts/Resource/CIDFont
Solaris	/opt/Acrobat3/Fonts	/opt/Acrobat3/Fonts/Resource/CMap	/opt/Acrobat3/Fonts/Resource/CIDFont
Windows	\Windows\Fonts	\Windows\Fonts\Resource\CMap	\Windows\Fonts\Resource\CIDFont

The environment variables listed in Table 11 must be exported or added so that the IBM Content Manager OnDemand PDF Indexer for Multiplatforms can locate DBCS font files.

Table 11. Environment Variables for Adobe DBCS Font Files

Environment Variable	Purpose
ACRO_RES_DIR	Location of the CMap files
PSRESOURCEPATH	Locations of the CIDFont, CMap, and DBCS font files

For AIX:

```
export ACRO_RES_DIR=/usr/lpp/Acrobat3/Fonts/Resource/CMap
export PSRESOURCEPATH=/usr/lpp/Acrobat3/Fonts:/usr/lpp/Acrobat3/Fonts/Resource/
CMap:/usr/lpp/Acrobat3/Fonts/Resource/CIDFont
```

For HP-UX:

```
export ACRO_RES_DIR=/opt/Acrobat3/Fonts/Resource/CMap
export PSRESOURCEPATH=/opt/Acrobat3/Fonts:/opt/Acrobat3/Fonts/Resource/
CMap:/opt/Acrobat3/Fonts/Resource/CIDFont
```

For Solaris:

```
export ACRO_RES_DIR=/opt/Acrobat3/Fonts/Resource/CMap
export PSRESOURCEPATH=/opt/Acrobat3/Fonts:/opt/Acrobat3/Fonts/Resource/
CMap:/opt/Acrobat3/Fonts/Resource/CIDFont
```

For Windows:

Add the System Variables listed in Table 12 to the Environment Variables dialog box.

Table 12. System Environment Variables for Adobe DBCS Font Files

Variable	Value
ACRO_RES_DIR	\Windows\Fonts\Resource\CMap
PSRESOURCEPATH	\Windows\Fonts;\Windows\Fonts\Resource\CMap;\Windows\Fonts\Resource\CIDFont

PDF indexing limitations

If you are using the IBM Content Manager OnDemand PDF Indexer for Multiplatforms to generate index data for PostScript and PDF files that are created by user-defined programs, you need to keep the following in mind:

- The IBM Content Manager OnDemand PDF Indexer for Multiplatforms can process files that are up to 4 GB in size.
- The IBM Content Manager OnDemand PDF Indexer for Multiplatforms supports DBCS languages. However, IBM does not provide any DBCS fonts. You can purchase DBCS fonts from Adobe. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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.

- To print PDF documents from the Content Manager OnDemand server, you must use the Content Manager OnDemand server print function. The server print function requires Infoprint.
- If a font is referenced in an input file but not embedded in the file and the IBM Content Manager OnDemand PDF Indexer for Multiplatforms cannot locate the font, Times New Roman is substituted in place of the referenced font. If the customer purchases additional fonts and installs them on the system, the additional fonts can be embedded at indexing time if they are referenced in an input file and the location is specified on the FONTLIB parameter. See “FONTLIB” on page 181 for more information.

Input data requirements

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms processes PDF input data. PostScript data generated by applications must be processed by Acrobat Distiller before you run the IBM Content Manager OnDemand PDF Indexer for Multiplatforms. The online documentation provided with Acrobat Distiller describes methods you can use to generate PDF data.

You can use several methods to provide the IBM Content Manager OnDemand PDF Indexer for Multiplatforms with access to input data, including FTP and NFS. If you use a file transfer method to copy PDF data to the Content Manager OnDemand server, you must transfer the files in binary format.

If you plan to automate the data indexing and loading process on the Content Manager OnDemand server by using the ARSLOAD program, the input file name must identify the application group and application to load. Use the following convention to name your input files:

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

Important: The .PDF file name extension is required to initiate a load process.

Unless you specify otherwise, the ARSLOAD program uses the FORM part of the file name to identify the application group to load. However, you can use the **-G** parameter to specify a different part of the file name (MVS, JOBNAME, or DATASET) to identify the application group to load.

If the application group contains more than one application, you must identify the application to load; otherwise the load will fail. You can run the ARSLOAD program with the **-A** parameter to specify the part of the input file name (MVS, JOBNAME, DATASET, or FORM) that identifies the application.

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

National language support for indexed PDF documents

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms supports DBCS languages. However, IBM does not provide any DBCS fonts. You can purchase DBCS fonts from Adobe. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms supports all DBCS fonts, except encrypted Japanese fonts. See “Specifying the location of Adobe fonts” on page 173 if you plan to use DBCS font files.

Data values that you specify on FIELD parameters must be encoded in the same code page as the document. For example, if the characters in the document are encoded in code page 500, any data values that you specify on FIELD parameters must be encoded in code page 500. Examples of data values that you might specify include FIELD default and constant values.

When loading data using the IBM Content Manager OnDemand PDF Indexer for Multiplatforms, the locale must be set appropriately for the code page of the documents. For example, if the code page of the documents is 954, set the locale environment variable to ja_JP or some other locale that correctly identifies upper and lower case characters in code page 954.

For more information about NLS in Content Manager OnDemand, see *IBM Content Manager OnDemand for Multiplatforms: Installation and Configuration Guide*.

Chapter 16. Parameter reference

This parameter reference assumes that you will use the ARSLOAD program to process your input files. When you use the ARSLOAD program to process input files, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms ignores any values that you may provide for the INDEXDD, INPUTDD, MSGDD, OUTPUTDD, RESOBJDD, and PARMDD parameters. If you run the ARSPDOCI program from the command prompt or call it from a user-defined program, then you must provide values for the INPUTDD, OUTPUTDD, and PARMDD parameters and verify that the default values for the INDEXDD, RESOBJDD, and MSGDD parameters are correct.

BOOKMARKS

Indicates whether to copy the bookmarks from the original document to the new documents. The default value is YES, which means that all the bookmarks from the original document are copied to each new document that is created by the IBM Content Manager OnDemand PDF Indexer for Multiplatforms. Many of these bookmarks might no longer be valid. If the original document contains many bookmarks, you can reduce the size of the new documents by not copying the bookmarks.

Required?

No

Default Value

YES

Syntax

BOOKMARKS=*value*

Options and values

The *value* can be:

YES

The bookmarks are copied to each new document that is created by the IBM Content Manager OnDemand PDF Indexer for Multiplatforms. This is the default value.

NO

The bookmarks are not copied.

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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms begins searching for the field, relative to a trigger or 0 (zero) for the same page as the trigger. If you specify BASE=0, the *page* value can be

-16 to 16. If you specify BASE=TRIGGER, the *page* value must be 0 (zero), which is relative to the sheet number where the trigger string value is located.

TRIGGER=*n*

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

BASE={0 | TRIGGER}

Determines whether the IBM Content Manager OnDemand PDF Indexer for Multiplatforms uses the upper left coordinates of the trigger string box to locate the field. Choose from 0 (zero) or TRIGGER. If BASE=0, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms adds zero to the field string box coordinates. If BASE=TRIGGER, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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=u1(4,4),lr(5,8),1,'Total'  
FIELD2=u1(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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms always locates the field in a one inch box, one inch to the right of the location of the trigger string value. If the IBM Content Manager OnDemand PDF Indexer for Multiplatforms finds the trigger string value in location u1(4,4),lr(5,5), it attempts to find the field in location u1(5,4),lr(6,5). If the IBM Content Manager OnDemand PDF Indexer for Multiplatforms finds the trigger string value in location u1(4,6),lr(5,7), it attempts to find the field in location u1(5,6),lr(6,7).

Note: A field that is based on the location of a trigger (BASE=TRIGGER) can be defined at any location on the page that contains the trigger. Previously, a field that was based on the location of a trigger had to be defined to the right and below the upper left point of the trigger. With this change, the x or y values can be negative, as long as the resulting absolute field coordinates of the field string rectangle are still in the range of 0 <= x <= 45 and 0 <= y <= 45. The u1(x,y) and lr(x,y) coordinates of the FIELD parameter are relative offsets from the u1(x,y) coordinates of the trigger. For example, suppose the field string rectangle is located at u1(1,1), lr(2,2) which is an absolute location on the page. If the trigger string rectangle is located at u1(5,5), lr(7,7), then the field coordinates would be u1(-4,-4), lr(-3,-3).

MASK='field_mask'

The pattern of symbols that the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms to match a 15-character alphabetic field, such as a name.

Matches numeric characters. For example:
MASK='#####'

Causes the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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.

Note: The string that you specify for the mask can contain any character. For example, given the following definitions:

```
TRIGGER2=*,25,'ACCOUNT' FIELD2=0,38,11,(TRIGGER=2,BASE=0,MASK='0000-####-#')
```

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms selects the field only if the data in the field columns contains an eleven-character string comprised of (in order) any letter, three zeros, a dash character, any four numbers, a dash character, and any number.

DEFAULT='value'

Defines the default index value, when there are no words within the coordinates provided for the field string box. You might specify the default value in hexadecimal. See an example in “Examples.”

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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms assigns the index associated with FIELD3 the value NOT AUDITED, if the field string box is blank.

Examples

The following field parameter causes the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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)
```

Hexadecimal default value:

```
TRIGGER1 = u1(4.5,1.25), lr(5.75,1.5), *, 'ACCOUNT'  
FIELD1   = u1(6.6,1.25), lr(7.1,1.25),0,(default=x'30313233')  
INDEX1   = 'Account',FIELD1,(TYPE=GROUP)
```

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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms does not validate the type or content of the constant. You might specify the constant value in hexadecimal. See an example in “Examples.”

Examples

The following field parameter causes the IBM Content Manager OnDemand PDF Indexer for Multiplatforms to store the same text string in each INDEX1 value it creates.

```
FIELD1='000000000'  
INDEX1='acct',FIELD1
```

The following field parameters cause the IBM Content Manager OnDemand PDF Indexer for Multiplatforms to concatenate a constant value with the index value extracted from the data. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms concatenates the constant value specified in the FIELD1 parameter to each index value located using the FIELD2 parameter. The concatenated string value is stored in the database. In this example, the account number field in the data is 14 bytes in length. However, the account number in the database is 19 bytes in length. Use a constant field to concatenate a constant five byte prefix (0000-) to all account numbers extracted from the data.

```
FIELD1='0000-'  
FIELD2=ul(2,2),lr(2.5,2.25),0,(TRIGGER=1,BASE=0)  
INDEX1='acct_num',FIELD1,FIELD2
```

Hexadecimal constant field:

```
FIELD1 = X'4D524830303252'  
FIELD2 = ul(6.6,1.25), lr(7.1,1.25),0,(default=x'30313233')  
INDEX1 = 'Account',FIELD1,FIELD2,(TYPE=GROUP)
```

You can combine a hexadecimal value and a value that is extracted from the document in an index:

```
FIELD1 = X'4D524830303252'  
FIELD2 = ul(6.0,1.4), lr(7.2,1.75),0  
INDEX1 = 'Account',FIELD1,FIELD2,(TYPE=GROUP)
```

Related parameters

INDEX parameter on page 182.

TRIGGER parameter on page 189.

FONTLIB

Identifies the directory or directories in which fonts are stored. Specify any valid path. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms searches for fonts in the order that the paths are listed. If a font is referenced in an input file but not embedded in the file, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms attempts to locate the font in the directory or directories listed on the FONTLIB parameter. If the font is located, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms adds it to the output

file. If a font is referenced in an input file and the IBM Content Manager OnDemand PDF Indexer for Multiplatforms cannot locate the font, the referenced font is substituted by using one of the base Adobe Type 1 fonts that are provided by IBM. If the customer purchases additional fonts and installs them on the system, the additional fonts can be embedded at indexing time if they are referenced in an input file and are present in one of the directories specified on the FONTLIB parameter.

Required?

No

Default Value

/usr/lpp/Acrobat3/Fonts (AIX)

/opt/Acrobat3/Fonts (HP-UX, Sun Solaris)

\PSFONTS (Windows)

Syntax

FONTLIB=*pathlist*

Options and values

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

```
FONTLIB=/usr/lpp/Acrobat3/Fonts:/usr/lpp/ars/fontlib
```

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms searches the paths in the order in which they are specified. Delimit path names in UNIX with the colon (:) character. Delimit path names in Windows with the semicolon (;) character.

INDEX

Identifies the index name and the field or fields on which the index is based. You must specify at least one index parameter. You can specify up to 32 index parameters. When you create index parameters, IBM 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 *acct_num* would be a meaningful index name. The index value of INDEX1 would be an actual account number, for example, 000123456789.

The index name is a string from 1 to 250 bytes in length. IBM recommends that you name the index the same as the application group database field name.

FIELD_{*nn*}

The name of the field parameter or parameters that the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms to create group-level indexes for date index values (the IBM Content Manager OnDemand PDF Indexer for Multiplatforms supports group-level indexes only). When the index value changes, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms closes the current group and begins a new group.

```
INDEX1='report_date',FIELD1
```

The following index parameters cause the IBM Content Manager OnDemand PDF Indexer for Multiplatforms to create group-level indexes for customer name and account number index values. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms closes the current group and begins a new group when either the customer name or the account number index value changes.

```
INDEX1='name',FIELD1  
INDEX2='acct_num',FIELD2
```

Related parameters

FIELD parameter on page 178.

INDEXDD

Specifies the name or the full path name of the index object file. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms writes indexing information to the index object file. If you specify the file name without a path, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms puts the index object file in the current directory. If you do not specify the INDEXDD parameter, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms writes indexing information to the file INDEX.

Required?

No

Note: When you process input files with the ARSLOAD program, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms ignores any value that you may specify for the INDEXDD parameter. (The ARSLOAD program sets the file name for the IBM Content Manager OnDemand PDF Indexer for Multiplatforms.) 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=*filename*

Options and values

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

Notes:

1. Filenames and path names are case sensitive in UNIX, but not in Windows.
2. If you do not specify the INDEXDD parameter, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms writes the index object file to the current directory.

INDEXMODE

Determines whether the IBM Content Manager OnDemand PDF Indexer for Multiplatforms uses metadata indexes instead of triggers, fields, and indexes. If not specified, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms uses the trigger, field, and index parameters to perform the indexing.

Required?

No

Default Value

<none>

Note: If INDEXMODE is specified along with TRIGGER, FIELD, or INDEX parameters, they are ignored.

Syntax

INDEXMODE=*mode*

Options and values

The *mode* can be:

METADATA - Use metadata indexes

Examples

The following parameters cause the IBM Content Manager OnDemand PDF Indexer for Multiplatforms to extract metadata indexes and create a resource file. No other parameters are required.

```
RESTYPE=ALL  
INDEXMODE=METADATA
```

INDEXSTARTBY

Determines the page number by which the IBM Content Manager OnDemand PDF Indexer for Multiplatforms must locate the first group (document) within the input file. 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 file 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms must locate the first group on the first page of the input file. This parameter is helpful if the input file contains header pages. For example, if the input file contains two header pages, you can specify a page number one greater than the number of header pages (INDEXSTARTBY=3) so that the IBM Content Manager OnDemand PDF Indexer for Multiplatforms will stop processing only if it does not locate the first group by the third page in the input data.

Important: When you use INDEXSTARTBY to skip header pages, ACIF does not copy the non-indexed pages to the output file. For example, if you specify INDEXSTARTBY=3 and ACIF finds the first index on page three, then ACIF skips pages one and two. Page three is the first page in the output file.

Required?

No

Default Value

1

Syntax

INDEXSTARTBY=*value*

Options and values

The *value* is the page number by which the IBM Content Manager OnDemand PDF Indexer for Multiplatforms must locate the first group (document) in the input file.

INPUTDD

Specifies the name or the full path name of the PDF input file that the IBM Content Manager OnDemand PDF Indexer for Multiplatforms will process.

Required?

No

Note: When you process input files with the ARSLOAD program, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms ignores any value that you may supply for the INPUTDD parameter. If you process input files with the ARSPDOCI program, then you must specify a value for the INPUTDD parameter.

Default Value
<none>

Syntax

INPUTDD=*name*

Options and values

The *name* is the file name or full path name of the input file. On UNIX servers, file and path names are case sensitive. If you specify the file name without a path, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms searches the current directory for the specified file.

MSGDD

Specifies the name or the full path name of the file to which the IBM Content Manager OnDemand PDF Indexer for Multiplatforms writes error messages. If you do not specify the MSGDD parameter, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms writes messages to standard error (UNIX) or the console (Windows).

Required?
No

Note: When you process input files with the ARSLOAD program, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms ignores any value that you may supply for the MSGDD parameter. If you process input files with the ARSPDOCI program, then verify the value of the MSGDD parameter.

Default Value
stderr (UNIX)
console (Windows)

Syntax

MSGDD=*name*

Options and values

The *name* is the file name or full path name of the file to which the IBM Content Manager OnDemand PDF Indexer for Multiplatforms writes error messages. On UNIX servers, file and path names are case sensitive. If you specify the file name without a path, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms writes the error file to the current directory.

OUTPUTDD

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

Required?
No

Note: When you process input files with the ARSLOAD program, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms ignores any value that you may supply for the OUTPUTDD parameter. If you process input files with the ARSPDOCI program, then you must specify a value for the OUTPUTDD parameter.

Default Value

<none>

Syntax

OUTPUTDD=*name*

Options and values

The *name* is the file name or full path name of the output file. On UNIX servers, file and path names are case sensitive. If you specify the file name without a path, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms writes the output file to the current directory.

PARMDD

Specifies the name or the full path name of the file that contains the indexing parameters that are used to process the input data.

Required?

No

Note: When you process input files with the ARSLOAD program, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms ignores any value that you may supply for the PARMDD parameter. If you process input files with the ARSPDOCI program, then you must specify a value for the PARMDD parameter.

Default Value

<none>

Syntax

PARMDD=*name*

Options and values

The *name* is the file name or full path name of the file that contains the indexing parameters. On UNIX servers, file and path names are case sensitive. If you specify the file name without a path, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms searches for the file in the current directory.

REMOVERES

Indicates whether or not to remove unused resources before the indexer collects resources and creates the indexes. The input file is examined and a new copy is saved in the Content Manager OnDemand temporary directory. This new copy is then used for processing, and the original input file is not changed. You can change the location of the temporary directory by specifying the PDF parameter TEMPDIR. Ensure that the temporary directory has enough space to hold the file. If a file contains many unused resources, you can greatly reduce the size of the resource file and speed up the indexing process by using this parameter. If a file

does not contain any unused resources, then do not specify this parameter. You can use this parameter without resource collection.

Required?

No

Default Value

NO

Syntax

REMOVERES=*value*

Options and values

The *value* can be one of the following:

- YES** The unused resources are removed before the indexer collects resources (if requested) and creates the indexes.
- NO** The unused resources are not removed before the indexer collects resources (if requested) and creates the indexes.

RESOBJDD

Specifies the name or the full path name of the resource object file. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms collects resources to the resource object file. If you specify the file name without a path, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms puts the resource object file in the current directory. Use the RESOBJDD parameter in conjunction with the RESTYPE parameter for the IBM Content Manager OnDemand PDF Indexer for Multiplatforms to collect resources.

Required?

No

When you process input files with the ARSLOAD program, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms ignores any value that you might supply for the RESOBJDD parameter. If you process input files with the ARSPDOCI program and want to collect resources, then you must specify a value for the RESOBJDD parameter.

Default Value

<none>

Syntax

RESOBJDD=*filename*

Options and values

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

Important:

1. File names and path names are case-sensitive on AIX, Solaris, HP-UX, and Linux, but are not case-sensitive on Windows.
2. If the PDF file does not contain resources, no RESOBJDD file is produced.

RESTYPE

Determines the types of PDF print resources that the IBM Content Manager OnDemand PDF Indexer for Multiplatforms should collect and include in the resource group file.

Required?

No

Default Value

NONE

Syntax

RESTYPE={ *NONE* | *ALL* | [*FONT*] [,*IMAGE*] }

Options and values

NONE

No resource file is created.

ALL All fonts and images are collected in the resource file.

FONT Fonts are collected in the resource file.

IMAGE

Images are collected in the resource file.

TEMPDIR

Specifies the name of the directory that the IBM Content Manager OnDemand PDF Indexer for Multiplatforms uses for temporary work space.

Required?

No

Default Value

/arstmp (UNIX)

C:\TEMP (Windows)

Syntax

TEMPDIR=*directory*

Options and values

The *directory* is a valid directory name.

TRACEDD parameter

For more information on the TRACEDD parameter, see Chapter 20, "Trace facility," on page 199.

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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms begins searching on the first page in the input file. The IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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. If the trigger is represented by a double byte or Unicode font in the document, enter the trigger string in hexadecimal. You can use hexadecimal and non-hexadecimal triggers together. See “Examples” on page 191 for a hexadecimal trigger.

Examples

TRIGGER1

The following TRIGGER1 parameter causes the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms to attempt to match the string value Account Number within the coordinates provided for the trigger string box. The trigger can be found on the same page as TRIGGER1.

```
TRIGGER2=ul(1,2.25),lr(2,2.5),0, 'Account Number'
```

The following trigger parameter causes the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms adds the upper left coordinates of the trigger string box to the coordinates provided for the field. The trigger can be found on the page following TRIGGER1.

```
TRIGGER2=ul(4,4),lr(5,8),1, 'Total '  
FIELD2=ul(1,0),lr(2,1),0, (TRIGGER=2,BASE=TRIGGER)
```

Hexadecimal trigger

This example shows how to code a trigger that represents two side-by-side UTF-8 characters in a document. In this example, each UTF-8 character consists of three bytes. Do not code the index name in hexadecimal.

```
TRIGGER1=UL(1.54,5.40),LR(1.79,5.53),*,X'E6AC8AE79B8A'  
FIELD1=UL(2.29,3.86),LR(3.34,4.04),0, (TRIGGER=1,BASE=0)  
INDEX1='emp_name',FIELD1,(TYPE=GROUP)
```

In this example, hexadecimal and non-hexadecimal triggers are used together:

```
TRIGGER1=UL(6.49,1.72),LR(6.89,1.93),*,X'E8BD8920E7A7BB'  
TRIGGER2=UL(7.02,2.34),LR(7.53,2.60),0, 'Page 1'
```

Related parameters

The FIELD parameter on page 178.

Chapter 17. Message reference

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms detects a number of error conditions that can be logically grouped into several categories:

- **Informational**

When the IBM Content Manager OnDemand PDF Indexer for Multiplatforms processes a file, it issues informational messages that allow the user to determine if the correct processing parameters have been specified. These messages can assist in providing an audit trail.

- **Warning**

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms issues a warning message and a return code of 4 (four) when the fidelity of the document may be in question.

- **Error**

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms issues an error message and return code of 8 (eight) or 16 (sixteen) and terminates processing the current input file. Most error conditions detected by the IBM Content Manager OnDemand PDF Indexer for Multiplatforms fall into this category. The exact method of termination may vary. For certain severe errors, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms may fail with a segment fault. This is generally the case when some system service fails. In other cases, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms terminates with the appropriate error messages written either to standard error or to a file. When the IBM Content Manager OnDemand PDF Indexer for Multiplatforms is invoked by the ARSLOAD program, error messages are automatically written to the system log. If you run the ARSPDOCI command, you can specify the name or the full path name of the file to hold the processing messages by using the **MSGDD** parameter.

- **Adobe Toolkit**

If the Adobe libraries fail to initialize, the IBM Content Manager OnDemand PDF Indexer for Multiplatforms issues an error message with a return code of 16 and stops processing the current input file.

- **Internal Error**

The IBM Content Manager OnDemand PDF Indexer for Multiplatforms issues an error message and return code of 16 (sixteen) and terminates processing the current input file.

Note: See *IBM Content Manager OnDemand: Messages and Codes* for a list of the messages that can be generated by the IBM Content Manager OnDemand PDF Indexer for Multiplatforms, along with explanations of the messages and actions that you can take to respond to the messages. The messages that are generated by the IBM Content Manager OnDemand PDF Indexer for Multiplatforms are listed in the Common Server section of the messages publication.

Chapter 18. ARSPDOCI reference

Purpose

Generate index data for a PDF file.

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, “ARSPDUMP reference,” on page 197 for more information.

Syntax

Note: The following syntax should be used only when you run the ARSPDOCI program from the command line or call it from a user-defined program.

```
➤➤ ARSPDOCI [BOOKMARKS=value] [COORDINATES=metric] FIELDn=spec ➤➤
➤ [FONTLIB=pathList] INDEXn=spec [INDEXDD=fileName] ➤
➤ [INDEXSTARTBY=pageNumber] INPUTDD=fileName [MSGDD=fileName] ➤
➤ OUTPUTDD=fileName [PARMDD=fileName] [RESOBJDD=fileName] ➤
➤ [RESTYPE={ NONE | ALL | [FONT] [,IMAGE] }] [TEMPDIR=fileSystem] ➤
➤ TRIGGERn=spec ➤➤
```

Description

The ARSPDOCI program can be used to index a PDF file. The ARSLOAD program automatically calls the ARSPDOCI program if the input data type is PDF and the indexer is PDF. If you need to index a PDF file and you do not want to use the ARSLOAD program to process the file, then you can run the ARSPDOCI program from the command line or call it from a user-defined program.

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

If a font is referenced in an input file but not embedded in the file and the IBM Content Manager OnDemand PDF Indexer for Multiplatforms cannot locate the font, the referenced font is substituted by using one of the base Adobe Type 1 fonts that are provided by IBM. If the customer purchases additional fonts and installs them on the system, the additional fonts can be embedded at indexing time if they are referenced in an input file and the location is specified on the FONTLIB parameter. See “FONTLIB” on page 181 for more information.

Parameters

Refer to Chapter 16, “Parameter reference,” on page 177 for details about the parameters that you can specify when you run the ARSPDOCI program from the command line or a user-defined program.

Files

/usr/lpp/ars/bin/arspdoci

The AIX executable program.

/opt/ondemand/bin/arspdoci

The HP-UX and Solaris executable program.

\Program Files\IBM\OnDemand for Windows\bin\arspdoci

The Windows executable program.

Chapter 19. ARSPDUMP reference

Purpose

Print the locations of text strings on a page.

The ARSPDUMP program lists the locations of text strings on a page in a PDF file. 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, “ARSPDOCI reference,” on page 195 for more information.

Syntax

```
➤➤ ARSPDUMP -f inputFile [-F fontFile] [-o outputFile]
➤ -p sheetNumber [-t tempDir]
```

Description

The ARSPDUMP program can be used to identify the locations of text strings on a page in a PDF file.

The output of the ARSPDUMP program contains a list of the text strings on the page and the coordinates for each string.

If a font is referenced in a PDF file, but not embedded, then the ARSPDUMP program attempts to find the font using information provided with the **-F** parameter. If the ARSPDUMP program does not find the font, then it uses a substitute Adobe Type 1 font.

Parameters

-f inputFile

The file name or full path name of the PDF file to process. On UNIX servers, file and path names are case sensitive.

-F fontDir

Identifies directories in which fonts are stored. Specify any valid path. On UNIX servers, use the colon (:) character to separate path names. On Windows servers, use the semicolon (;) character to separate path names. The ARSPDUMP program searches the paths in the order in which they are specified. If you do not specify this flag and name a font directory, then the ARSPDUMP program attempts to locate fonts in the /usr/lpp/Acrobat3/Fonts directory (AIX), /opt/Acrobat3/Fonts directory (HP-UX, Sun Solaris), or \PSFONTS directory (Windows).

-o outputFile

The file name or full path name of the file into which the ARSPDUMP

program writes output messages. On UNIX servers, file and path names are case sensitive. If you do not specify this flag and name a file, then the ARSPDUMP program writes output to stdout (UNIX) or the console (Windows).

-p sheetNumber

The number of the page in the PDF file that you want the ARSPDUMP program to process. This is the page that contains the text strings that you want to use to define triggers and fields. The sheet number is the order of the page as it appears in the file, beginning with the number 1 (one), for the first page in the file. Contrast with page identifier, which is user-defined information that identifies each page (for example, iv, 5, and 17-3).

-t tempDir

Identifies the directory that the ARSPDUMP program uses for temporary work space. Specify any valid directory name. If you do not specify this flag and name a directory, then the ARSPDUMP program uses the /arstmp directory (UNIX servers) or the C:\TMP directory (Windows servers) for temporary work space.

Examples

1. The following example shows how to invoke the ARSPDUMP program to print the strings and locations of text found on page number one of sample.pdf to sample.out:

```
arspdump -f sample.pdf -o sample.out -p 1
```

2. The following example shows how to invoke the ARSPDUMP program to print the strings and locations of text found on page number three of sample.pdf to sample.out:

```
arspdump -f sample.pdf -o sample.out -p 3
```

Files

/usr/lpp/ars/bin/arspdump

The AIX executable program.

/opt/ondemand/bin/arspdump

The HP-UX and Sun Solaris executable program.

\Program Files\IBM\OnDemand for Windows\bin\arspdump

The Windows executable program.

Chapter 20. Trace facility

The tracing capability for the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms, the following is required:

- The parameter file, which specifies the fields, triggers, indexes and other indexing information
- The PDF input file to process
- The trace parameters `tracedd` and `tracelevel`

Note: See Chapter 16, “Parameter reference,” on page 177 for information about the trace parameters.

The parameter file and PDF input file can be processed by running the IBM Content Manager OnDemand PDF Indexer for Multiplatforms from the command line. For example:

```
arspdoci parmdd=filen.parms inputdd=filen.pdf outputdd=filen.out indexdd=filen.ind  
tracedd=filen.trace tracelevel=15
```

Where:

`arspdoci` is the name of the command-line version of the IBM Content Manager OnDemand PDF Indexer for Multiplatforms 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 IBM Content Manager OnDemand PDF Indexer for Multiplatforms

`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

`tracelevel`= specifies the amount of detail to be included in the trace

Note: See Chapter 18, “ARSPDOCI reference,” on page 195 for more information about the parameters that may be specified when running the ARSPDOCI program.

After running the IBM Content Manager OnDemand PDF Indexer for Multiplatforms with the trace, the output file specified by the `tracedd`= parameter will contain detailed information about the processing that took place and where the IBM Content Manager OnDemand PDF Indexer for Multiplatforms is failing during the process.

Part 4. Xenos transforms

This part 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.

This part is organized into the following sections:

- General information. See Chapter 21, “Understanding Xenos,” on page 203, Chapter 22, “Xenos transforms,” on page 205, and Chapter 23, “Loading data,” on page 209.
- Information about indexing, transforming, and loading input files into the system. See Chapter 24, “How to specify parameters to the Xenos transforms,” on page 211 and Chapter 25, “JS program reference,” on page 215.

Chapter 21. Understanding Xenos

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 transforms 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 program, license, and documentation from your IBM representative. Your IBM representative can also provide education 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 documents.

The Xenos transforms can be used in an Content Manager OnDemand system when loading input files into the system.

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.

Figure 37 on page 204 shows a high-level overview of how the Xenos transforms fit 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, then you can transform AFP and Metacode documents that are stored in the system before sending them to the Web browser. See *IBM Content Manager OnDemand for Multiplatforms: Web*

Enablement Kit Implementation Guide for more information.

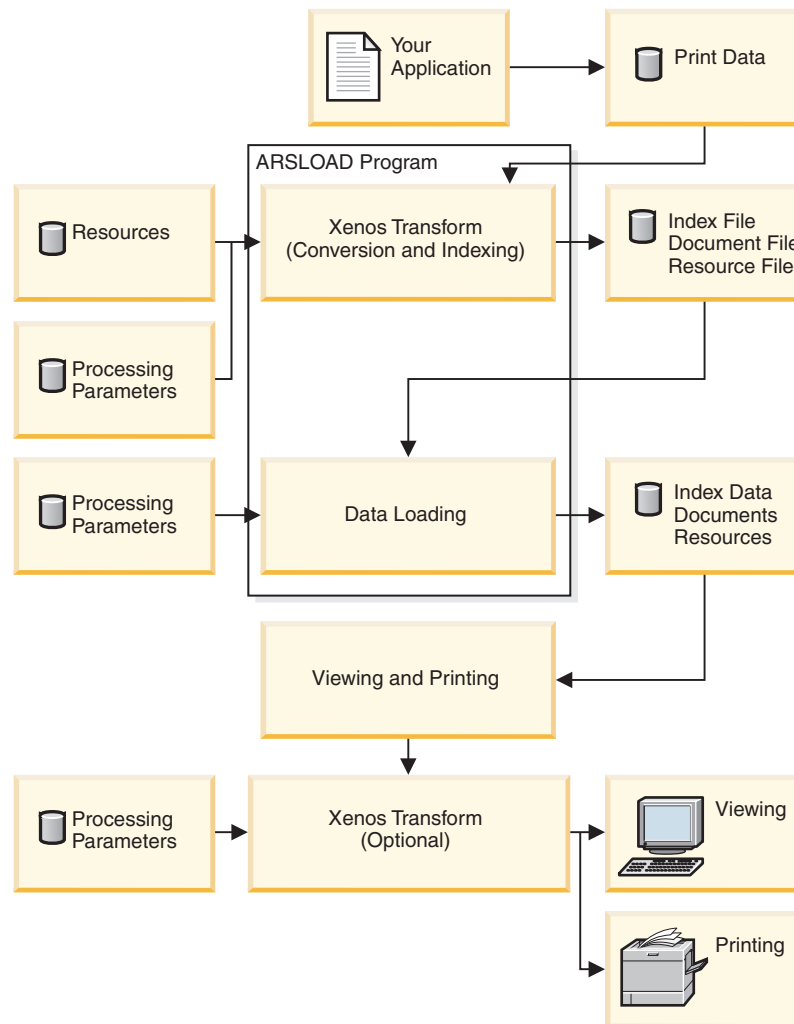


Figure 37. How the Xenos Transform fits into the OnDemand Environment

Chapter 22. Xenos transforms

You can use the Xenos transforms 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 during load processing:

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

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. 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 Indexer Information tab to Xenos. Specify processing parameters on the Indexer Information tab.

Metacode to AFP

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 Indexer Information tab to Xenos. Specify processing parameters on the Indexer Information tab.

Metacode to Metacode

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 Indexer Information tab to Xenos. Specify processing parameters on the Indexer Information tab.

Metacode to PDF

When loading Metacode/DJDE 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. 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 Indexer Information tab to Xenos. Specify processing parameters on the Indexer Information tab.

PCL to PDF

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. 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 Indexer Information tab to Xenos. Specify processing parameters on the Indexer Information tab.

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 38 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.

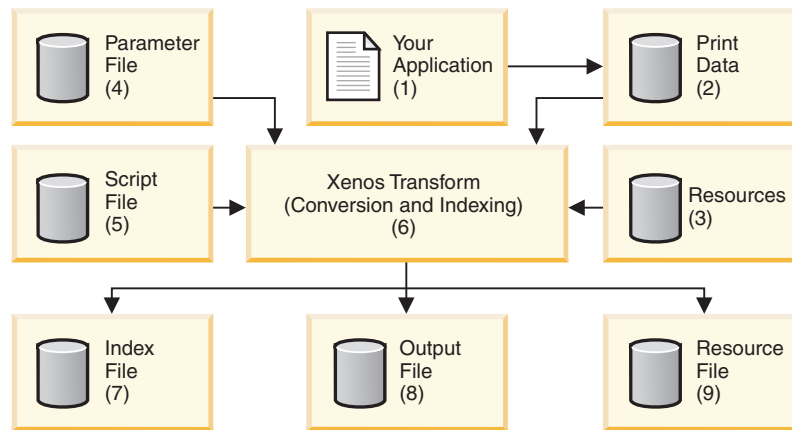


Figure 38. Using the Xenos Transforms to Prepare Files for Loading and Viewing

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 46 on page 223 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 47 on page 225 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 Content Manager OnDemand generic index format.
8. The Output File contains the indexed groups of pages, also known as documents in Content Manager 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 39 shows the second part of the load process – the ARSLOAD program processes the files that were created by the Xenos transform.

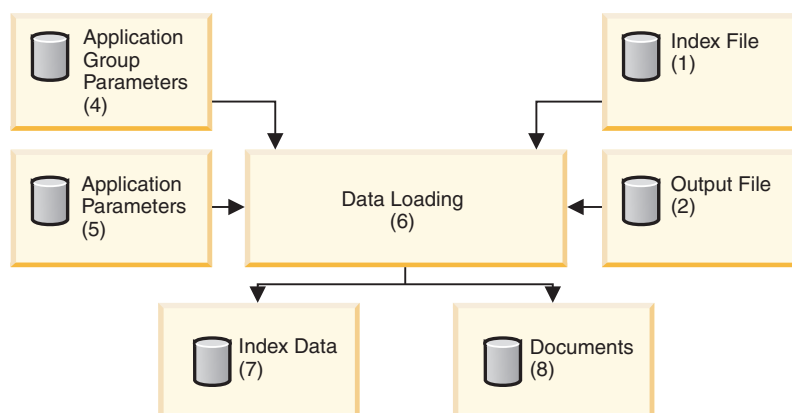


Figure 39. Loading Data into OnDemand

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 Indexer Information tab, 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 212 for information about creating the application parameters. Figure 40 on page 214 and Figure 41 on page 214 show examples of the Indexer Information tab.
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. How to specify parameters to the Xenos transforms

This section describes how to specify parameters to the Xenos transforms to assist you with developing indexing parameters and to specify the parameters that are used by the ARSLOAD program and the Xenos transforms to index and convert input files and load them into the system. See *IBM Content Manager OnDemand for Multiplatforms: Web Enablement Kit Implementation Guide* for information about how to specify parameters when you use ODWEK 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 212.

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 JS program) to help you determine the location of the text strings in the input data. The JS 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 JS 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 a page” on page 218 for an example of how to run the JS 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=c:\tmp\dmlic_IBM_NT.txt
```

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=c:\tmp\sample.par
```

Figure 40 on page 214 shows an example of the Indexer Information tab 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 = 'c:\program files\xenos\dmsl.lib'
scriptvar = ( 'Parser', 'META' )
scriptvar = ( 'Generator', 'PDF' )
scriptvar = ( 'NumberOfFields', 2 )
.
.
.
OnDemandXenosParmEnd
```

Figure 41 on page 214 shows an example of the Indexer Information tab 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=c:\tmp\sample.dms
```

Figure 40 on page 214 shows an example of the Indexer 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_SetPart( par_h, 'fdinput', inputfile )
.
.
.
OnDemandXenosScriptEnd
```

Figure 41 on page 214 shows an example of the Indexer 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 222 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 that is specified in the indexing script (in the example, dm_DASDWrite(index_h, "CODEPAGE:850"). For example, if CODEPAGE:954 is specified, set the locale environment variable to ja_JP or some other locale that correctly identifies upper and lower case characters in code page 954.

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 40. 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 41. Application Indexing Information – Specifying Details

Chapter 25. JS program reference

Purpose

The JS program is the main Xenos transform program. You can use the JS program in the following ways:

- Manually run the JS program to print the locations of the text strings found on the pages of an input file.
- Automatically call the JS program from the ARSLOAD program to process an input file, generate an OnDemand generic index file and the other output files, convert the input data, and load the data into the system.

The JS program can process input files that contain AFP, Metacode, or PCL data.

Note: 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 211 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 212.

Syntax

Note: The following shows the syntax that is used by the ARSLOAD program to call the JS program.

Important: All of the parameters are required.

```
►►JS—parms=—fileName—report=—fileName—scriptvar=inputfile=—fileName—►
►--scriptvar=outputfile=—fileName—scriptvar=indexfile=—fileName—►
►--scriptvar=resourcefile=—fileName—license=—fileName—►►
```

Description

The JS program can be used to convert AFP, Metacode/DJDE, and PCL print input data. The JS 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 JS 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 JS 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 use the JS 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 JS 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.

- Convert input data, generate an index file, and for Metacode output, collect resources. The ARSLOAD program will automatically call the JS program if the Indexer that is specified on the Indexer Information page for the application is Xenos. You can also run the JS program from the command line to generate an index file manually.

Parameters

Note: The following shows the parameters that are used by the ARSLOAD program to call the JS program.

Important: All of the parameters are required.

-parms=

Use to specify the file name or full path name of the file that contains the names of the parameter file and the script file. The parameter file contains the parameters that are used to index and convert the input data. The script file creates the OnDemand generic index file and the other output files. See Figure 44 on page 219 and Figure 46 on page 223 for examples of parameter files. See Figure 45 on page 220 and Figure 47 on page 225 for examples of script files. On UNIX servers, file and path names are case sensitive.

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

-report=

Use to specify the file name or full path 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. On UNIX servers, file and path names are case sensitive.

-scriptvar=inputfile=

Use to specify the file name or full path name of the file that contains the input data to process. On UNIX servers, file and path names are case sensitive.

-scriptvar=outputfile=

Use to specify the file name or full path name of the file that will contain the converted output data. On UNIX servers, file and path names are case sensitive.

Note: If you are using the JS program to generate the text strings and locations, then you can discard the output file.

-scriptvar=indexfile=

Use to specify the file name or full path name of the file that will contain the index data that will be loaded into the OnDemand database. On UNIX servers, file and path names are case sensitive.

-scriptvar=resourcefile=

When generating Metacode output, specifies the file name or full path name of the file that will contain the resources that were collected. On UNIX servers, file and path names are case sensitive.

-license=

Use to specify the file name or full path name of the file that contains the license information required to run the Xenos programs. On UNIX servers, file and path names are case sensitive.

Note: You must obtain a valid license file from Xenos.

Examples

The syntax of the JS program is the same, whether you use the program to print the text strings and the location information or you use the program to generate the index and output files to be loaded into OnDemand. The only variables in the process are the type of conversion to run and the indexing parameters that the JS program uses to process the input data, which are specified in the processing parameter file, and the script file.

Figure 42 shows an example of how to run the JS program.

```
js -parms="c:\program files\xenos\parms.auto"
   -report="c:\program files\xenos\sample.rep"
   -scriptvar=inputfile="c:\program files\xenos\sample.mta"
   -scriptvar=outputfile="c:\program files\xenos\sample.out"
   -scriptvar=indexfile="c:\program files\xenos\sample.ind"
   -scriptvar=resourcefile="c:\program files\xenos\sample.res"
   -license="c:\program files\xenos\dmlic_IBM_NT.txt"
```

Figure 42. Running the JS Program

In Figure 42:

- The file that contains the name of the parameter file and the script file is `parms.auto`
- The file that the job report will be written to is `sample.rep`
- The file that contains the input data to process is `sample.mta`
- The file that will contain the converted data to be loaded into the system is `sample.out`
- The file that will contain the index data that will be loaded into the database is `sample.ind`
- The file that will contain the resources that will be loaded into the system is `sample.res`
- The file that contains the Xenos license information is `dmlic_IBM_NT.txt`

Figure 43 shows the contents of the example `parms.auto` file.

```
fdjobparm='\\program files\xenos\meta2pdf\sample.par'
fddmscript='\\program files\xenos\meta2pdf\sample.dms'
```

Figure 43. Sample Parameter File

In Figure 43, `sample.par` is the name of the file that contains the processing parameters and `sample.dms` is the name of the script file.

Generating the locations of text strings on a page

This example shows how to use the JS 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 JS 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 222).

- Figure 44 on page 219 shows the processing parameters. 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 and the script variable Generator represents the output file format. Table 13 lists the possible values for the Parser and Generator variables when loading input files into the system.

Table 13. 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 JS 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 JS program will process the first eleven pages of the input file. **Note:** The input data must contain the information that the JS 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 JS to generate the text strings and the locations.
- Figure 45 on page 220 shows the script file. **Note:** The script file does not create an index file, because an index file is not needed for this step.


```

/* Sample processing parameters for Meta2PDF transform */

JS:

/* DM Script Library - XG supplied functions */
fddmslib = 'c:\program files\xenos\dmsl.lib'

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

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      = 'c:\program files\xenos\resources\meta\%s.fnt'
fdjsl      = 'c:\program files\xenos\resources\meta\%s.jsl'
respectshift = NO

/* MTA2PDF Options */
fdfonttab  = 'c:\program files\xenos\meta2pdf\newfont.tab'

```

Figure 44. Sample Parameters for the Xenos Transform (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)

METADL-DLIST:

parmdpi    = 300
pagefilter = all
resfilter  = all
fddloutput = sample.dl

fieldname  = (PAGE)
fieldword  = (20, and, %20)
fieldphrase = (%500, OneSpace)
fieldpara  = (%500)

```

Figure 44. Sample Parameters for the Xenos Transform (Part 2 of 2)

```

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}'outputfile);

/* initialize */
file_open = FALSE

dlpage = dm_GetDLPage(gen_h);

do while(dlpage <> 'EOF')
  if file_open = FALSE then do
    if(generator = 'AFP') then do
      rc = dm_AFPGenOpen(gen_h, '{GROUPFILEENTRY}'outputfile);
    end
    else if(generator = 'META') then do
      rc = dm_METAGenOpen(gen_h, '{GROUPFILEENTRY}'outputfile);
    end
    else if(generator = 'PDF') then do
      rc = dm_PDFGenOpen(gen_h, '{GROUPFILEENTRY}'outputfile);
    end
    file_open = TRUE
  end
end

```

Figure 45. Sample JS Program Script File (Part 1 of 2)

```

    if(generator = 'AFP') then do
        rc = dm_AFPGenWrite(gen_h, dlpage );
    end
    else if(generator = 'META') then do
        rc = dm_METAGenWrite(gen_h, dlpage );
    end
    else if(generator = 'PDF') then do
        rc = dm_PDFGenWrite(gen_h, dlpage );
    end

    dlpage = dm_GetDLPage(par_h);
end

if file_open = TRUE then do
    if(generator = 'AFP') then do
        rc = dm_AFPGenClose( gen_h )
    end
    else if(generator = 'META') then do
        rc = dm_METAGenClose( gen_h )
    end
    else if(generator = 'PDF') then do
        rc = dm_PDFGenClose(gen_h );
    end
end

rc = dm_DASDClose( dasd_h )
return;

```

Figure 45. Sample JS Program Script File (Part 2 of 2)

Indexing and converting input data

This example shows how to use the JS program to process a Metacode/DJDE input file and produce an index file and a converted output file that can be loaded into the system.

- Figure 46 on page 223 shows the processing parameters. 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 and locations of the index fields, using the text string coordinates that were generated in "Generating the locations of text strings on a page" on page 218.
 3. 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.
 4. The `fieldbase` parameters identify the locations of index fields using the specified offset from the anchor string.
 5. The `fieldname` parameters identify the names of the index fields.
 6. 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 JS 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.
 7. The absence of the `FDDLOUTPUT` parameter means that the JS program will not generate the text strings and the location information.
 8. For information regarding the `FieldLocate`, `FieldName`, and `FieldBase` parameters, refer to your Xenos documentation.
- Figure 47 on page 225 shows the script file. **Note:** The script file inserts a line delimiter at the end of each line that is written to the index file. The example script file uses the line delimiter for Windows systems – `X'0D0A'` (Carriage Return and Line Feed characters). For UNIX systems, the line delimiter is `X'0A'` (Line Feed character).
 - Figure 48 on page 229 shows the generic index file that was created by the script file.

```

/* Sample indexing parameters for META2PDF transform */

JS:

/* DM Script Library - XG supplied functions */
fddmslib = 'c:\program files\xenos\dmsl.lib'

scriptvar = ('Parser', 'META')
scriptvar = ('Generator', 'PDF')
scriptvar = ('NumberOfFields', 2)
scriptvar = ('Field.1', 'Name')
scriptvar = ('Field.2', 'Policy')

METADL-METAP:

/* Metacode Parser Options */
startjdl   = GTIJDL
startjde   = START
startpage  = 0
stoppage   = 0
position   = WORD
native     = NO
cc         = YES
shade      = NONE
dashline   = NO
ldmethod   = LARGEST
cctran     = NONE

/* File Defs */
fdfnt      = 'c:\program files\xenos\resources\meta\%s.fnt'
fdjsl      = 'c:\program files\xenos\resources\meta\%s.jsl'
respectshift = NO

```

Figure 46. Sample Parameters for the Xenos Transform (Part 1 of 2)

```

/* MTA2PDF Options */
fdfonttab = 'c:\program files\xenos\meta2pdf\newfont.tab'

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)

METADL-DLIST:

parmdpi     = 300
pagefilter  = all
resfilter   = all

fieldname   = (PAGE)
fieldword    = (20, and, %20)
fieldphrase  = (%500, OneSpace)
fieldpara    = (%500)

/* field 1 - extract name */
fieldlocate = ('InsName', 'Insured')
fieldname   = ('Name')
fieldbase    = ('InsName', +275,
               '=' , -35,
               '=' , +800,
               '=' , +30)

/* field 2 - extract policy number */
fieldname    = ('Policy')
fieldbase     = ('InsName', +300,
                '=' , +100,
                '=' , +800,
                '=' , +170)

```

Figure 46. Sample Parameters for the Xenos Transform (Part 2 of 2)

```

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}'outputfile);
rc = dm_DASDOpen(index_h, indexfile );

/* initialize */
do i = 1 to NumberOfFields
    fieldvaluesave.i = ""
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"||crlf);
rc = dm_DASDWrite(index_h, "COMMENT: File generated by the xenos process"||crlf);
dt = DATE('N');
ts = TIME('N');
rc = dm_DASDWrite(index_h, "COMMENT:" dt||" "||ts||crlf);
rc = dm_DASDWrite(index_h, "COMMENT:"||crlf);
rc = dm_DASDWrite(index_h, "CODEPAGE:819"||crlf||crlf);

dlpage = dm_GetDLPage(gen_h);

do while(dlpage <> 'EOF')
    if file_open = FALSE then do
        if(generator = 'AFP') then do
            rc = dm_AFPGenOpen(gen_h, '{GROUPFILEENTRY}'outputfile);
        end
        else if(generator = 'META') then do
            rc = dm_METAGenOpen(gen_h, '{GROUPFILEENTRY}'outputfile);
        end
        else if(generator = 'PDF') then do
            rc = dm_PDFFGenOpen(gen_h, '{GROUPFILEENTRY}'outputfile);
        end
    end

```

Figure 47. Sample JS Program Script File (Part 1 of 4)

```

    if rc = 0 then do
        file_open = TRUE
    end
    do i = 1 to NumberOfFields
        fieldvalue.i = dm_GetText( dlpge, 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
                    docbreak = 1
                end
            end
        end
    end
    if docbreak = 1 then do
        if(generator = 'AFP') then do
            rc = dm_AFPGenClose( gen_h )
        end
        else if(generator = 'META') then do
            rc = dm_METAGenClose( gen_h )
        end
        else if(generator = 'PDF') then do
            rc = dm_PDFFGenClose( 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 || crlf )
            field_value = "GROUP_FIELD_VALUE:" || fieldvaluesave.i
            rc = dm_DASDWrite( index_h, field_value || crlf )
        end
    end

```

Figure 47. Sample JS Program Script File (Part 2 of 4)


```

/* 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 || crlf)
group_length = "GROUP_LENGTH:" || length
rc = dm_DASDWrite( index_h, group_length || crlf)
group_filename = "GROUP_FILENAME:" || outputfile
rc = dm_DASDWrite( index_h, group_filename || crlf || crlf)

if(generator = 'AFP') then do
  rc = dm_AFPGenOpen(gen_h, '{GROUPFILEENTRY}'outputfile);
end
else if(generator = 'META') then do
  rc = dm_METAGenOpen(gen_h, '{GROUPFILEENTRY}'outputfile);
end
else if(generator = 'PDF') then do
  rc = dm_PDFGenOpen(gen_h, '{GROUPFILEENTRY}'outputfile);
end
if rc = 0 then do
  file_open = TRUE
end
end

if(generator = 'AFP') then do
  rc = dm_AFPGenWrite(gen_h, d1page );
end
else if(generator = 'META') then do
  rc = dm_METAGenWrite(gen_h, d1page );
end
end

```

Figure 47. Sample JS Program Script File (Part 3 of 4)

```

else if(generator = 'PDF') then do
    rc = dm_PDFGenWrite(gen_h, dlpage );
end
dlpage = dm_GetDLPage(par_h);
end

if file_open = TRUE then do
    if(generator = 'AFP') then do
        rc = dm_AFPGenClose( gen_h )
    end
    else if(generator = 'META') then do
        rc = dm_METAGenClose( gen_h )
    end
    else if(generator = 'PDF') then do
        rc = dm_PDFGenClose( 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 || crlf)
    field_value = "GROUP_FIELD_VALUE:" || fieldvaluesave.i
    rc = dm_DASDWrite( index_h, field_value || crlf)
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 || crlf)
group_length = "GROUP_LENGTH:" || length
rc = dm_DASDWrite( index_h, group_length || crlf)
group_filename = "GROUP_FILENAME:" || outputfile
rc = dm_DASDWrite( index_h, group_filename || crlf)

rc = dm_DASDClose( dasd_h )
rc = dm_DASDClose( index_h )
return;

```

Figure 47. Sample JS Program Script File (Part 4 of 4)

```

COMMENT: OnDemand Generic Index File Format
COMMENT: This file has been generated by the xenos process
COMMENT: 20 Feb 2001 14:16:18
COMMENT:
CODEPAGE:819

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:111783
GROUP_FILENAME:\program files\xenos\meta2pdf\sample.out
GROUP_FIELD_NAME:Name
GROUP_FIELD_VALUE:WARD T. JONES, MD
GROUP_FIELD_NAME:Policy
GROUP_FIELD_VALUE:0030-334-33
GROUP_OFFSET:111783
GROUP_LENGTH:339119
GROUP_FILENAME:\program files\xenos\meta2pdf\sample.out
GROUP_FIELD_NAME:Name
GROUP_FIELD_VALUE:Mike R. Smith.
GROUP_FIELD_NAME:Policy
GROUP_FIELD_VALUE:0333-888-45
GROUP_OFFSET:450902
GROUP_LENGTH:106689
GROUP_FILENAME:\program files\xenos\meta2pdf\sample.out
GROUP_FIELD_NAME:Name
GROUP_FIELD_VALUE:MIKE R. SMITH
GROUP_FIELD_NAME:Policy
GROUP_FIELD_VALUE:0333-888-45
GROUP_OFFSET:557591
GROUP_LENGTH:338985
GROUP_FILENAME:\program files\xenos\meta2pdf\sample.out
GROUP_FIELD_NAME:Name
GROUP_FIELD_VALUE:Barbara L. Schuster
GROUP_FIELD_NAME:Policy
GROUP_FIELD_VALUE:4567-001-77
GROUP_OFFSET:896576
GROUP_LENGTH:106705
GROUP_FILENAME:\program files\xenos\meta2pdf\sample.out
GROUP_FIELD_NAME:Name
GROUP_FIELD_VALUE:BARBARA L. SCHUSTER
GROUP_FIELD_NAME:Policy
GROUP_FIELD_VALUE:4567-001-77
GROUP_OFFSET:1003281
GROUP_LENGTH:334110
GROUP_FILENAME:\program files\xenos\meta2pdf\sample.out

```

Figure 48. Generic Index File Produced by the JS Program Script

Loading Metacode documents in large object format

Large Object documents are stored in groups of pages called Segments, which can be retrieved individually to improve performance. The System Administrator sets the number of pages in a Segment in the Application. In order to load documents in Large Object format, the loading process must create an index file that contains, in addition to the group information, offset and length information for each page in the document. Since OnDemand uses Xenos to create the index file when Metacode documents are loaded, the Xenos script must be written to create the index file with the necessary information.

The following is an example of an index file for loading Metacode documents in Large Object format. The index file describes one group, identified by the group index named Acct with a value of 4488131010005300. Therefore, one hit will appear in the hit list when this folder is searched.

```
COMMENT: OnDemand Generic Index File Format
COMMENT: This file has been generated by the xenos process
COMMENT: 02 Mar 2004 11:32:15
COMMENT:
CODEPAGE:819

PAGE_OFFSET:0
PAGE_LENGTH:99
PAGE_IDENTIFIER:1

PAGE_OFFSET:99
PAGE_LENGTH:7
PAGE_IDENTIFIER:2

PAGE_OFFSET:106
PAGE_LENGTH:1397
PAGE_IDENTIFIER:3

PAGE_OFFSET:1503
PAGE_LENGTH:1103
PAGE_IDENTIFIER:4

PAGE_OFFSET:2606
PAGE_LENGTH:893
PAGE_IDENTIFIER:5

PAGE_OFFSET:3499
PAGE_LENGTH:4054
PAGE_IDENTIFIER:6

PAGE_OFFSET:7553
PAGE_LENGTH:1397
PAGE_IDENTIFIER:7

PAGE_OFFSET:8950
PAGE_LENGTH:1103
PAGE_IDENTIFIER:8
```

Figure 49. Index File for Loading Metacode Documents in Large Object Format (Part 1 of 2)

```

.
.
.

PAGE_OFFSET:203740
PAGE_LENGTH:1113
PAGE_IDENTIFIER:100

GROUP_FIELD_NAME:Acct
GROUP_FIELD_VALUE:4488131010005300
GROUP_OFFSET:0
GROUP_LENGTH:204853
GROUP_FILENAME:/home/paulaj/documorph/mbna-1o/mbna.out

```

Figure 49. Index File for Loading Metacode Documents in Large Object Format (Part 2 of 2)

The following is an example of a Xenos script file that creates an index file that can be used to load documents in Large Object format. The script file created the index file shown above.

```

/* this script file calculates page offsets from the beginning of the file */
/* and creates a new group when the account number changes */

TRUE = 1;
FALSE = 0;

call dm_initialize

par_h = dm_StartParser("Meta");
gen_h = dm_StartGenerator("Meta");

rc = dm_SetParm(par_h, 'fdinput', '{LF2xm}'inputfile);

rc = dm_SetParm(par_h, 'fdfnt', '{IORES-OUT:'resourcefile'}/home/paulaj/documorp
h/mbna/res/%s.fnt');
rc = dm_SetParm(par_h, 'fdfrm', '{IORES-OUT:'resourcefile'}/home/paulaj/documorp
h/mbna/res/%s.frm');
rc = dm_SetParm(par_h, 'fdimg', '{IORES-OUT:'resourcefile'}/home/paulaj/documorp
h/mbna/res/%s.img');
rc = dm_SetParm(par_h, 'fdjsl', '{IORES-OUT:'resourcefile'}/home/paulaj/documorp
h/mbna/res/%s.jsl');

```

Figure 50. Script File that Creates an Index File for Loading Documents in Large Object Format (Part 1 of 5)

```

/* 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}'outputfile);
rc = dm_DASDOpen(index_h, indexfile );

test = dm_SetParm(par_h, "DJDECOMMENTS", "YES")

file_open = FALSE
save_PageBytesWritten = 0
save_GroupBytesWritten = 0
page_id = 0
crlf = '0a'X
first_time = TRUE;

/* 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:819"||crlf );

acct_num_save = ""

dlpage = dm_GetDLPage(par_h);

```

Figure 50. Script File that Creates an Index File for Loading Documents in Large Object Format (Part 2 of 5)

```

do while(dlpage \= 'EOF')
  if file_open = FALSE then do
    rc = dm_METAGenOpen(gen_h, '{GROUPFILEENTRY}{LF2xm}'outputfile);
    file_open = TRUE
  end

  new_doc_tf = FALSE;

  text = dm_FindComment(dlpage, "DB", "CONTAINS", "FIRST");
  IF (text<>'') THEN DO;
    acct_num = STRIP(SUBSTR(text,1,16));
    if acct_num_save = "" then
      acct_num_save = acct_num;
    new_doc_tf = TRUE;
  END;

  text = dm_FindComment(dlpage, "PM", "CONTAINS", "FIRST");
  IF (text<>'') THEN DO;
    acct_num = STRIP(SUBSTR(text,1,16));
    if acct_num_save = "" then
      acct_num_save = acct_num;
    new_doc_tf = TRUE;
  END;

  IF first_time = TRUE THEN DO;
    new_doc_tf = FALSE;
  END;

```

Figure 50. Script File that Creates an Index File for Loading Documents in Large Object Format (Part 3 of 5)

```

IF new_doc_tf THEN DO;
    rc = dm_METAGenClose( gen_h )
    file_open = FALSE

    rc = write_page_index()

    /* write out index values to the index file */
    field_name = "GROUP_FIELD_NAME:Acct"
    rc = dm_DASDWrite( index_h, field_name )
    field_value = "GROUP_FIELD_VALUE:"acct_num_save
    rc = dm_DASDWrite( index_h, field_value )
    acct_num_save = acct_num;

    rc = write_group_index()

    rc = dm_METAGenOpen(gen_h, '{GROUPFILEENTRY}{LF2xm}'outputfile);

    file_open = TRUE
end /* end if new_doc_tf TRUE */
else
do
    first_time = FALSE;
    rc = dm_DASDSize(dasd_h)
    if dm_size > 0 then
        rc = write_page_index()
    end

    rc = dm_METAGenWrite(gen_h, dlpag );

    dlpag = dm_GetDLPage(par_h);
end

if file_open = TRUE then do
    rc = dm_METAGenClose( gen_h )
end

rc = write_page_index()

```

Figure 50. Script File that Creates an Index File for Loading Documents in Large Object Format (Part 4 of 5)

```

/* write out final index values to the index file */
field_name = "GROUP_FIELD_NAME:Acct"
rc = dm_DASDWrite( index_h, field_name )
field_value = "GROUP_FIELD_VALUE:acct_num"
rc = dm_DASDWrite( index_h, field_value )
rc = write_group_index()

rc = dm_DASDClose( dasd_h )
rc = dm_DASDClose( index_h )
return;

write_group_index:
    rc = dm_DASDSize(dasd_h)
    TotalBytesWritten = dm_size
    length = TotalBytesWritten - save_GroupBytesWritten
    offset = TotalBytesWritten - length
    save_GroupBytesWritten = TotalBytesWritten
    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 || crlf )
    return rc;

write_page_index:
    rc = dm_DASDSize(dasd_h)
    TotalBytesWritten = dm_size
    page_length = TotalBytesWritten - save_PageBytesWritten
    page_offset = save_PageBytesWritten
    save_PageBytesWritten = save_PageBytesWritten + page_length

    page_offset_line = "PAGE_OFFSET:" || page_offset
    rc = dm_DASDWrite( index_h, page_offset_line )
    page_length_line = "PAGE_LENGTH:" || page_length
    rc = dm_DASDWrite( index_h, page_length_line )
    page_id = page_id + 1
    page_id_line = "PAGE_IDENTIFIER:" || page_id
    rc = dm_DASDWrite( index_h, page_id_line || crlf )
    return rc;

```

Figure 50. Script File that Creates an Index File for Loading Documents in Large Object Format (Part 5 of 5)

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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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. The sample

programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol ([®] or [™]), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml

Adobe, Acrobat, Portable Document Format (PDF), and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds 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.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Portions of the OnDemand Windows client program contain licensed software from Pixel Translations Incorporated, © Pixel Translations Incorporated 1990, 2003. All rights reserved.

Other company, product or service names may be trademarks or service marks of others.

Index

A

accessing reports 15, 22, 32, 44

ACIF

about 3

ACIF input record exit 103, 104

AFP resources 8, 55, 63, 64, 78, 79, 81, 82, 83, 84, 86, 94

apka2e input record exit 103

asciinp 103

asciinp input record exit 103

asciinpe 104

asciinpe input record exit 104

CC parameter 49, 113, 114

CCTYPE parameter 49, 113, 114

CHARS parameter 50

constant field 58

CONVERT parameter 51

CONVERT requirement 51

CPGID parameter 52

DCFPAGENAMES parameter 52

default index value 57

EBCDIC data 10

error messages 97

examples 15

exit 103

exit, index 104

exit, input file 100

exit, output record 107

exit, resource retrieval 109

exit, user programming 99

extended options 53

EXTENSIONS parameter 53

FDEFLIB parameter 55

field mask 60

FIELD parameter 56

fields 19, 28, 39, 56

FILEFORMAT parameter 62

FONTLIB parameter 63

FORMDEF parameter 64, 113, 114

group indexes 5, 65, 67

group-level indexes 9

GROUPMAXPAGES parameter 65

GROUPNAME parameter 66

IMAGEOUT parameter 66

index exits 104

INDEX parameter 67

index, exit 100

INDEXDD parameter 70

indexes 20, 29, 41, 67

indexing 100

indexing parameter reference 49

indexing parameters

about 5

example of 17, 25, 34, 46

INDEXOBJ parameter 71

INDEXOBJ requirement 71

INDEXSTARTBY parameter 72

INDEXEXIT parameter 73

INPEXIT parameter 73

input exits 100

input file, exit 100

input record exit 103, 104

ACIF (continued)

input record exits 103

input, user exit 99

INPUTDD parameter 74

INSERTIMM parameter 74

introduction 3

Invoke Medium Map structured field 122

invoking program to index input file 141

JCL statement defined 139

large object support 71

line data 7

LINECNT parameter 75

mask 95

mask option 60

MCF2REF parameter 76

message file 139

messages 97

MSGDD parameter 76

NEWPAGE parameter 77

non-zero return codes 113

OS/390 requirements 4

OUTEXIT parameter 77

output file format 136

output record exits 107

OUTPUTDD parameter 78

overview 3

OVLYLIB parameter 78

page indexes 51, 67, 71

page-level indexes 3, 10

PAGEDEF parameter 79, 113, 114, 115

pagerange indexes 69

parameter file 139

parameter reference 49

parameters

about 5

example of 17, 25, 34, 46

parameters syntax 141

parameters, z/OS 140

PARMDD parameter 80

PDEFLIB parameter 81

print file attributes 113

print file attributes, CC parameter 113, 114

print file attributes, CCTYPE parameter 113, 114

print file attributes, FORMDEF parameter 113, 114

print file attributes, PAGEDEF parameter 113, 114, 115

print file attributes, PRMODE parameter 113, 114

print file attributes, TRC parameter 113, 114

PRMODE parameter 81, 113, 114

program requirements 4, 139

PSEGLIB parameter 82, 94

reference 1

requirements 4, 139

RESEXIT parameter 83

RESFILE parameter 84

RESLIB parameter 84

RESOBJDD parameter 85

resource provided with ACIF 109

resource retrieval 109

resources 55, 63, 64, 78, 79, 81, 82, 83, 84, 86, 94

RESTYPE parameter 86

return code, non-zero 113

- ACIF (*continued*)
 - syntax rules, z/OS 140
 - TLE structured fields 44
 - TRACE parameter 88
 - transaction field 20, 59
 - TRC parameter 89, 113, 114
 - TRIGGER parameter 89
 - triggers 19, 27, 37, 89
 - UNIQUEBNGS parameter 93
 - usage 15
 - user exit input 99
 - user exit provided with ACIF 99
 - user exit, print file attributes 113
 - user programming exit 99
 - USERLIB parameter 94
 - USERMASK parameter 95
 - USERPATH parameter 96
 - using 4, 139
 - z/OS JCL statement 139
 - z/OS requirements 139
 - ACIF command
 - See* ACIF
 - ACIF exit
 - See* exits
 - ACIF input record exits
 - See* input record exit
 - ACRO_RES_DIR environment variable 174
 - Adobe font requirements
 - PDF indexer 173
 - Adobe PDF documents
 - See* PDF indexer
 - Advanced Function Printing (AFP)
 - See* AFP
 - AFP
 - about 7
 - converting line data to 7
 - converting with Xenos transforms 201
 - example of converting line data to 22, 32
 - example of indexing 44
 - fonts 63, 86
 - form definitions 55, 64
 - generic indexer, processing with 148
 - IMM structured fields 74
 - indexing 44
 - indexing with the generic indexer 148
 - line data
 - converting 7
 - MCF2 structured fields 76
 - overlays 78
 - page definitions 79, 81
 - page segments 82
 - processing with the generic indexer 148
 - processing with Xenos transforms 201
 - resources 55, 63, 64, 78, 79, 81, 82, 83, 84, 86, 94
 - collection 8
 - Set Coded Font Local structured fields 81
 - SOSI 81
 - user-defined resources 94
 - AFP API
 - See* AFP Application Programming Interface
 - AFP Application Programming Interface
 - Tag Logical Element 125
 - AFP Conversion and Indexing Facility (ACIF)
 - See* ACIF
 - AFPDS
 - converting to AFP data 7
 - placing TLEs in named groups 117
 - AFPDS (*continued*)
 - printing 118
 - transferring 122
 - AFPDS (AFP data stream)
 - defined 7
 - object support 7
 - resource support 7
 - ANSI carriage-control characters
 - See* carriage-control characters
 - apka2e exit 103
 - archiving, ACIF
 - indexing considerations 122
 - ARSACIF
 - See* ACIF
 - ARSPDOCI
 - COORDINATES parameter 177
 - error messages 193
 - FIELD parameter 178
 - FONTLIB parameter 181
 - INDEX parameter 182
 - INDEXDD parameter 183
 - INDEXMODE parameter 184
 - INDEXSTARTBY parameter 184
 - INPUTDD parameter 185
 - messages 193
 - MSGDD parameter 186
 - OUTPUTDD parameter 186
 - PARMDD parameter 187
 - reference 177, 195
 - TEMPDIR parameter 189
 - TRIGGER parameter 189
 - ARSPDOCI command
 - See* ARSPDOCI
 - ARSPDUMP program
 - reference 197
 - ASCII
 - parameter file for input data 13
 - ASCIINP exit
 - input record 103
 - ASCIINPE exit
 - input record 104
 - attributes
 - print file 113
- ## B
- BCOCA value 87
 - Begin Document Index structured field
 - defined 130
 - Begin Document structured field
 - defined 135
 - Begin Named Group structured field
 - defined 135
 - Begin Page structured field
 - defined 136
 - Begin Resource Group structured field
 - described 127
 - Begin Resource structured field
 - defined 127
 - bookmarks
 - PDF indexer 174
 - BOOKMARKS 177
 - BOX value 53
 - BTD
 - See* Begin Document structured field

C

- carriage controls 49, 75
- carriage-control characters
 - indexing considerations 123
- CC parameter
 - flags and values 49
 - print file attributes 113, 114
 - user exits 113, 114
- CCTYPE parameter
 - flags and values 49
 - print file attributes 113, 114
 - user exits 113, 114
- CELLED value 53
- CHARS parameter
 - flags and values 50
- CMS commands
 - invoking ACIF program to index input file 141
- code page
 - data indexing 52
 - generic indexer 151
 - IBM Content Manager OnDemand PDF Indexer for Multiplatforms 175
 - report file 52
 - Xenos indexer 213
- CODEPAGE: parameter 151
- commands
 - ARSPDOCI 195
 - ARSPDUMP 197
- COMMENT: parameter 152
- comments
 - in parameter file 141
- Composed Text Control (CTC) structured field
 - obsolete 136
- concatenation
 - z/OS files 143
- constant field 58, 180
- conversion 7, 51
- CONVERT parameter
 - flags and values 51, 74
- converting
 - AFP data to PDF 205
 - Metacode to AFP 205
 - Metacode to Metacode 205
 - Metacode to PDF 205
 - PCL to PDF 205
- converting line data to AFP 7
- coordinate system 164
- coordinates
 - on FIELD parameter for IBM Content Manager OnDemand PDF Indexer for Multiplatforms 178
 - on fieldbase parameter for Xenos transform 222
 - on TRIGGER parameter for IBM Content Manager OnDemand PDF Indexer for Multiplatforms 190
- COORDINATES parameter
 - flags and values 177, 184
- CPGID parameter
 - flags and values 52

D

- data
 - format 62
- DBCS font files
 - environment variables 174
 - where to install 173
- DBCS fonts 173, 175

- DCB requirements
 - message file, z/OS 140
 - output file, z/OS 140
- DCFPAGENAMES parameter
 - flags and values 52
- default index value
 - FIELD parameter option 57, 180
- defining fields 19, 28, 39
- defining indexes 20, 29, 41
- defining triggers 19, 27, 37
- diagnostic trace information 88
- document
 - DD statement for, z/OS 139
 - generic indexer parameter 153, 155
 - output format 133

E

- EBCDIC
 - parameter file for input data 13
- EBCDIC data
 - CCTYPE parameter 49
 - CPGID parameter 52
 - example of 10
 - indexing 10
 - specifying 10
 - TRIGGER parameter 89
 - USERMASK parameter 95
- EDI
 - See* End Document Index structured field
- EDT
 - See* End Document structured field
- EMPTYOK value 54
- End Document Index structured field
 - defined 132
- End Document structured field
 - defined 136
- End Named Group structured field
 - defined 136
- End Page structured field
 - defined 136
- End Resource Group structured field
 - defined 127
- End Resource structured field
 - defined 127
- ENG
 - See* End Named Group structured field
- environment variables
 - ACRO_RES_DIR 174
 - PSRESOURCEPATH 174
- EPG
 - See* End Page structured field
- ER
 - See* End Resource structured field
- ERG
 - See* End Resource Group structured field
- error messages
 - ACIF 97
 - ARSPDOCI program 193
 - PDF indexer 193
- examples 142, 144
 - AFP data, indexing 22, 32, 44
 - AFP document output formats 133
 - ASCII input data, parameter file for 13
 - EBCDIC input data, parameter file for 13
 - generic indexer 157
 - indexing 15, 22, 32, 44

- examples (*continued*)
 - invoking ACIF program to index input file 141
 - JCL and ACIF processing parameters 141
 - JS program 218, 222
 - line data, converting to AFP 22, 32
 - line data, indexing 15, 22, 32
 - print file attributes 113
 - Xenos transform 218, 222
 - z/OS JCL to invoke ACIF 139

- exits
 - apka2e 103
 - asciinp 103
 - asciinp 104
 - index 104
 - input 100
 - input record 103, 104
 - non-zero return codes 113
 - output 107
 - print file attributes provided 113
 - resource, provided with ACIF 109
- extended options 53
- EXTENSIONS parameter
 - flags and values 53
 - RESORDER value 87, 124

F

- FDEF value 86
- FDEFLIB parameter
 - flags and values 55
- FIELD parameter
 - constant field 58, 180
 - default index value 57, 180
 - flags and values 56, 178
 - mask option 60, 179
 - transaction field 59
 - trigger field 178
- fields
 - about 5
 - ACIF parameter 56
 - constant field 58, 180
 - default index value 57, 180
 - defining 19, 28, 39
 - generic indexer parameter 152, 153
 - IBM Content Manager OnDemand PDF Indexer for Multiplatforms parameter 178
 - mask option 60, 179
 - transaction field 20, 59
 - trigger field 178
 - Xenos transform 222
- file
 - message, ACIF 139
 - parameter, ACIF 139
- FILEFORMAT parameter
 - flags and values 62
- files
 - format 62
 - PDF indexer 175
- flags and values
 - REMOVERES 187
 - RESOBJDD 188
 - RESTYPE 189
- floating triggers 66, 91
- FONT value 87
- FONTLIB parameter
 - flags and values 63, 181

- fonts
 - CHARS parameter 50
 - converting 76
 - DBCS 173, 175
 - directory 63
 - library 63
 - location 63
 - Map Coded Font Format 2 structured fields 76
 - MCF2 structured fields 76
 - NLS 81, 173, 175
 - PDF indexer 173, 174, 181
 - resources 86
 - Set Coded Font Local structured fields 81
 - SOSI 81
 - specifying 50
 - TRCs 89
- form definitions 55, 64
- FORMDEF parameter
 - flags and values 64
 - print file attributes 113, 114
 - user exits 113, 114
- FRACLINE value 54

G

- generic indexer
 - about 145, 148
 - AFP data, processing 148
 - application group field names 152
 - code page 151
 - CODEPAGE: parameter 151
 - COMMENT: parameter 152
 - document 153, 155
 - examples 157
 - field names 152
 - field values 153
 - group indexes, defining 152, 153
 - GROUP_FIELD_NAME: parameter 152
 - GROUP_FIELD_VALUE: parameter 153
 - GROUP_FILENAME: parameter 153
 - GROUP_LENGTH: parameter 155
 - GROUP_OFFSET: parameter 155
 - input file 153, 155
 - introduction 145
 - national language support (NLS) 151
 - NLS 151
 - overview 145
 - parameter file 151, 157
 - using 145
- GOCA value 87
- graphical indexer
 - PDF input files 169
- group indexes
 - about 5
 - defining 67, 152, 182
 - defining for generic indexer 153
 - pages in a group 65
 - GROUP_FIELD_NAME: parameter 152
 - GROUP_FIELD_VALUE: parameter 153
 - GROUP_FILENAME: parameter 153
 - GROUP_LENGTH: parameter 155
 - GROUP_OFFSET: parameter 155
- group-level indexes
 - about 9
 - TLE structured fields 44
- GROUPMAXPAGES parameter
 - flags and values 65

GROUPNAME parameter
 flags and values 66
grouprange index 68

H

header pages
 skipping 72, 184

I

IBM Content Manager OnDemand PDF Indexer for
 Multiplatforms

 constant field 180
 default index value 180
 field mask 179
 fields 178
 graphical indexer 169
 group indexes 182
 indexes 182
 mask option 179
 resource collection 171
 trigger field 178
 triggers 189

IEL

See Index Element structured field

IMAGEOUT parameter

 flags and values 66

IMM structured fields 74

Index Element structured field

 considerations 122
 defined 130
 group-level 129
 index object file 122

index exit 104

index object file

 archiving considerations 122
 DD statement for, z/OS 140

INDEX parameter

 flags and values 67, 182
 JCL statement, z/OS 140
 z/OS, JCL statement 140

index user exit 73

INDEXDD parameter

 flags and values 70, 183

indexes

 about 5
 ACIF parameter 67
 defining 20, 29, 41
 generic indexer parameter 153
 group index 68
 grouprange index 68
 IBM Content Manager OnDemand PDF Indexer for
 Multiplatforms parameter 182
 page index 69
 pagerange index 69
 Xenos transform 222

indexing

 Adobe PDF documents 159
 AFP input files 201
 concepts 5
 constant field 58, 180
 CONVERT requirement 51
 default index value 57, 180
 EBCDIC data 10, 49, 52, 89, 95
 effect on document 133

indexing (*continued*)

 examples 15
 field mask 60, 179
 fields 5, 19, 28, 39, 56
 fields for IBM Content Manager OnDemand PDF Indexer
 for Multiplatforms 178
 file format 62
 floating triggers 91
 generic indexer 145
 graphical indexer 169
 group indexes 67, 182
 group-level indexes 9
 groups 5, 65
 header pages 72, 184
 helpful hints 122
 index exit 100
 indexes 5, 20, 29, 41, 67, 182
 INDEXOBJ requirement 71
 large object support 71
 line separator 62
 mask 95
 mask option 60, 179
 Metacode input files 201
 new line character 62
 page indexes 51, 67, 71
 page-level indexes 3, 10
 pagerange indexes 69
 parameters 5, 164
 PCL print files 201
 PDF indexer 159
 recordrange triggers 91
 reports
 example of 15
 skipping header pages 72, 184
 TLE structured fields 44
 transaction field 20, 59
 trigger field 178
 triggers 5, 19, 27, 37, 89, 189
 Xenos transform 206
 Xenos transforms 201
indexing parameters
 example of 17, 25, 34, 46
INDEXOBJ parameter
 flags and values 71
INDEXSTARTBY parameter
 flags and values 72, 184
INDEXEXIT parameter
 flags and values 73
inline resources
 processing 54, 55, 87, 124
INLINE value 87
INLONLY value 87
INPEXIT parameter
 flags and values 73
input
 z/OS 140
input file
 exit 100
 generic indexer parameter 153, 155
input record exit
 apka2e 103
input user exit 73
INPUTDD parameter
 flags and values 74, 185
Invoke Medium Map 74
 structured field 122
IOCA value 87

J

JCL

- ACIF JCL statement defined 139
- concatenating ACIF files, z/OS 144
- concatenation example, z/OS 144
- example, z/OS 142
- for ACIF job, z/OS 142
- for ACIF z/OS jobs 139
- for concatenating z/OS files 143
- invoking ACIF program to index input file 141
- OUTPUT JCL statement defined 140
- PRINTOUT JCL statement defined 139
- statement defined, ACIF JCL 139
- statement defined, OUTPUT JCL 140
- statement defined, PRINTOUT JCL 139
- z/OS example 142, 144

JS program

- examples 218, 222
- generic index file defined 229
- index file 229
- parameter file 219, 224
- reference 215
- script file defined 221, 228

L

- large object support 71

limitations

- PDF indexer 174

line data

AFP

- converting to 7, 22, 32
- converting to AFP 7, 22, 32
- defined 8
- example of indexing 15, 22, 32
- groups 65
- indexing 15, 22, 32
- pages in a group 65

line separator 62

LINECNT parameter

- flags and values 75

links

- PDF indexer 174

literal values

- determining how expressed 12

M

Map Coded Font Format 1 structured field

- converted 136

Map Coded Font Format 2 structured field

- archival, document integrity 136
- converting 76
- including fonts 87

mask 95

- FIELD parameter option 60, 179

maximum pages in a group 65

MCF-1

- See Map Coded Font Format 1 structured field

MCF-2

- See Map Coded Font Format 2 structured field

MCF2 structured fields 76, 87

MCF2REF parameter

- flags and values 76

message file

- DD statement for, z/OS 139

messages

- ACIF 97
- ARSPDOCI program 193
- PDF indexer 193

Metacode

- converting with Xenos transforms 201
- processing with Xenos transforms 201

metadata

- concepts 167
- indexing concepts 167

mixed-mode data

- defined 8

MO:DCA-P data stream

- ACIF changes to structured fields 7
- defined 7

MSGDD parameter

- flags and values 76, 186

multiple=up output

- page definition 122

N

naming input files

- PDF indexer 175

national language support (NLS)

- ACIF 52, 81
- generic indexer 151
- IBM Content Manager OnDemand PDF Indexer for Multiplatforms 173, 175
- Xenos indexer 213

new line separator 62

NEWPAGE parameter

- flags and values 77

NLS

- ACIF 52, 81
- generic indexer 151
- IBM Content Manager OnDemand PDF Indexer for Multiplatforms 173, 175
- Xenos indexer 213

non-zero return codes 113

O

OBJCON value 87

opening reports 18, 27, 36

OS/390

- ACIF requirements 4
- using ACIF 4

out-of-storage problem

- See Tag Logical Element structured field

OUTEXIT parameter

- flags and values 77

output file

- format 133

OUTPUT JCL statement

- defined, z/OS 140
- z/OS 140

output record exit 107

output user exit 77

OUTPUTDD parameter

- flags and values 78, 186

overlays 78

OVLY value 86

OVLYLIB parameter

- flags and values 78

P

- page definition
 - and resource file 137
 - multiple-up output 122
- page definitions 79, 81
- page indexes
 - about 134
 - CONVERT requirement 51
 - defining 67
 - INDEXOBJ requirement 71
 - large object support 71
- page segments 82
- page-level IELs 130
- page-level indexes
 - about 3, 10
 - TLE structured fields 44
- PAGEDEF parameter
 - flags and values 79
 - print file attributes 113, 114, 115
 - user exits 113, 114, 115
- pagerange index 69
- parameter file
 - ARSPDOCI program 177
 - ASCII input data, example of 13
 - comments 141
 - DD statement for, z/OS 139
 - EBCDIC input data, example of 13
 - generic indexer 157
 - PDF indexer 164, 177
 - syntax rules, z/OS 140
 - values spanning multiple records 141
- parameter values
 - spanning multiple records 141
- parameters
 - ARSPDOCI program 177, 195
 - ARSPDUMP program 197
 - CC 49
 - CCTYPE 49
 - CHARS 50
 - CODEPAGE: 151
 - COMMENT: 152
 - CONVERT 51
 - COORDINATES 177
 - CPGID 52
 - DCFAGENAMES 52
 - EXTENSIONS 53
 - FDEFLIB 55
 - FIELD 56, 178
 - FILEFORMAT 62
 - FONTLIB 63, 181
 - FORMDEF 64
 - generic indexer 151
 - GROUP_FIELD_NAME: 152
 - GROUP_FIELD_VALUE: 153
 - GROUP_FILENAME: 153
 - GROUP_LENGTH: 155
 - GROUP_OFFSET: 155
 - GROUPMAXPAGES 65
 - GROUPNAME 66
 - IMAGEOUT 66
 - INDEX 67, 182
 - INDEXDD 70, 183
 - indexing
 - example of 17, 25, 34, 46
 - INDEXMODE 184
 - INDEXOBJ 71
 - INDEXSTARTBY 72, 184
 - parameters (*continued*)
 - INDEXEXIT 73
 - INPEXIT 73
 - INPUTDD 74, 185
 - INSERTIMM 74
 - LINECNT 75
 - MCF2REF 76
 - MSGDD 76, 186
 - NEWPAGE 77
 - OUTEXIT 77
 - OUTPUTDD 78, 186
 - OVLYLIB 78
 - PAGEDEF 79
 - PARMDD 80, 187
 - PDEFLIB 81
 - PDF indexer 164, 177
 - PRMODE 81
 - PSEGLIB 82
 - RESEXIT 83
 - RESFILE 84
 - RESLIB 84
 - RESOBJDD 85
 - RESTYPE 86
 - TEMPDIR 189
 - TRACE 88
 - TRC 89
 - TRIGGER 89, 189
 - UNIQUEBNGS 93
 - USERLIB 94
 - USERMASK 95
 - USERPATH 96
 - Xenos transform 211
 - z/OS 140
 - PARMDD parameter
 - flags and values 80, 187
 - PCL print files
 - converting with Xenos transforms 201
 - processing with Xenos transforms 201
 - PDEFLIB parameter
 - flags and values 81
 - PDF indexer
 - about 159
 - Adobe font requirements 173
 - ARSPDOCI reference 195
 - ARSPDUMP reference 197
 - bookmarks 174
 - code page 175
 - concepts 164
 - coordinate system 164
 - DBCS fonts 173, 175
 - error messages 193
 - file naming conventions 175
 - font requirements 173
 - fonts 174, 181
 - indexing concepts 164
 - introduction 159
 - limitations 174
 - links 174
 - messages 193
 - naming input files 175
 - national language support (NLS) 173, 175
 - NLS 173, 175
 - overview 159
 - parameter file 164
 - parameter reference 177
 - printing 174
 - requirements 173

- PDF indexer (*continued*)
 - restrictions 174
 - support for DBCS fonts 174
 - system limitations 174
 - system requirements 173
 - transferring input files to 175
 - using 159
 - x, y coordinate system 164
- PDF resource collection 171
- Portable Document Format (PDF)
 - See* PDF indexer
- PostScript file
 - how processed by PDF indexer 175
 - relation to PDF file 162
- PostScript Passthrough markers
 - PDF indexer limitations 174
- PRCOLOR value 55
- print file attributes 113
 - CC parameter 113, 114
 - cctype parameters 113, 114
 - example 114
 - formdef parameters 113, 114
 - pagedef parameters 113, 114, 115
 - parameter, cc 113, 114
 - parameters, cctype 113, 114
 - parameters, formdef 113, 114
 - parameters, pagedef 113, 114, 115
 - parameters, prmode 113, 114
 - parameters, trc 113, 114
 - prmode parameters 113, 114
 - TRC parameters 113, 114
 - user exits 113, 114
- printing
 - PDF indexer 174
- PRINTOUT JCL statement
 - defined 139
- PRMODE parameter
 - flags and values 81
 - print file attributes 113, 114
 - user exits 113, 114
- PSEG value 86
- PSEGLIB parameter
 - flags and values 82
- PSRESOURCEPATH environment variable 174

R

- recordrange triggers 91
- REGION size for ACIF 139
- REMOVERES 187
- reports
 - accessing 15, 22, 32, 44
 - example of 15, 22, 32, 44
 - format 62
 - indexing 15
 - opening 18, 27, 36
- requirements
 - Adobe font requirements 173
 - fonts 173
 - PDF indexer 173
- RESEXIT parameter
 - flags and values 83, 84
- RESLIB parameter
 - flags and values 84
- RESOBJDD 188
- RESOBJDD parameter
 - flags and values 85

- RESOBJDD statement
 - z/OS 140
- RESORDER value 54, 55, 87
- resource
 - collection 8
- resource exit
 - provided with ACIF 109
- resource file
 - DD statement for, z/OS 140
 - format 126
- resource retrieval
 - file format 126
 - resource exit 109
- resource user exit 83
- resources
 - about AFP resources 8
 - directory 84
 - exits 83
 - file 84
 - fonts 63
 - form definitions 55, 64
 - group 84
 - inline, processing 54, 55, 87, 124
 - library 84
 - location 84
 - overlays 78
 - page definitions 79, 81
 - page segments 82
 - RESTYPE parameter 86
 - types of 86
 - user-defined 94
 - Xenos transform 207
- restrictions
 - PDF indexer 174
- RESTYPE 189
- RESTYPE parameter
 - flags and values 86

S

- separator pages
 - application-generated 123
 - removal from output 123
- Set Coded Font Local structured field 81
- skipping header pages 72, 184
- software
 - requirements for PDF indexer 173
- SOSI 81
- SPCMPRS value 55
- storage problem
 - See* Tag Logical Element structured field
- structured fields
 - Begin Document 135
 - Begin Document Index 130
 - Begin Named Group 133, 135
 - Begin Page 136
 - Begin Resource 127
 - Begin Resource Group 127
 - Composed Text Control (obsolete) 136
 - End Document 136
 - End Document Index 132
 - End Named Group 133, 136
 - End Page 136
 - End Resource 127
 - End Resource Group 127
 - group level 129
 - Index Element 122, 129, 130

- structured fields (*continued*)
 - Invoke Medium Map 74, 122
 - Map Coded Font Format 1 136
 - Map Coded Font Format 2 76, 136
 - page level 129
 - Presentation Text Data Descriptor 137
 - Set Coded Font Local 81
 - Tag Logical Element 122, 125, 131, 133, 135
- SYSIN JCL statement
 - z/OS 140
- SYSPRINT JCL statement
 - z/OS 140
- system requirements
 - Adobe font requirements 173
 - fonts 173
 - PDF indexer 173

T

- Tag Logical Element structured field
 - as part of the indexing process 131, 135
 - created in the output document file 133
 - defined 131
 - examples and rules 125
 - in named groups 117
 - out-of-storage problem, possible cause 117
 - storage problem, possible cause 117
- TEMPDIR parameter
 - flags and values 189
- TLE
 - See* Tag Logical Element structured field
- TRACE parameter
 - flags and values 88
- transaction field 20, 59
- translation reference characters (TRC) 50, 89
- TRC 50, 89
- TRC parameter
 - flags and values 89
 - print file attributes 113, 114
 - user exits 113, 114
- trigger field 178
- TRIGGER parameter
 - options and values 89, 189
- triggers
 - about 5
 - ACIF parameter 89
 - defining 19, 27, 37
 - floating 91
 - floating and groupname 66
 - IBM Content Manager OnDemand PDF Indexer for Multiplatforms parameter 189
 - recordrange 91
 - Xenos transform 222

U

- unformatted ASCII data
 - ACIF formatting of 8
 - defined 8
 - indexing 100
- UNIQUEBNGS parameter
 - flags and values 93
- user exits
 - index 73, 104
 - input 73, 99
 - output 77

- user exits (*continued*)
 - output record 107
 - print file attributes 113
 - provided with ACIF 99
 - resource 83
 - resource, provided with ACIF 109
- user programming exit 99
- USERAPPL
 - z/OS statement 139
- USERLIB parameter
 - flags and values 94
- USERMASK parameter
 - flags and values 95, 96
- using ACIF
 - in the z/OS environment 139

X

- x,y coordinate system 164
- Xenos transform
 - about 201
 - AFP to PDF 205
 - code page 213
 - examples 218, 222
 - generic index file defined 229
 - index file 229
 - indexing 206
 - introduction 201
 - JS program reference 215
 - loading data 209
 - Metacode to AFP 205
 - Metacode to Metacode 205
 - Metacode to PDF 205
 - national language support (NLS) 213
 - NLS 213
 - overview 201
 - parameter file 219, 224
 - parameters 211
 - PCL to PDF 205
 - resources 207
 - script file defined 221, 228
 - using 201

Z

- z/OS
 - ACIF parameters 140
 - ACIF requirements 139
 - concatenation example 144
 - DD statement for document file 139
 - INDEX JCL statement 140
 - index object file 140
 - input 140
 - invoking ACIF 139
 - JCL example 142, 144
 - JCL for ACIF job 139
 - JCL statement 139
 - JCL to invoke ACIF 139
 - message file, ACIF 139
 - OUTPUT JCL statement 140
 - parameters, ACIF 140
 - RESOBJ statement 140
 - SYSIN JCL statement 140
 - SYSPRINT JCL statement 140
 - USERAPPL statement 139
 - using ACIF 139



Program Number: 5724-J33

SC19-2952-00

