Content Manager OnDemand
Version 10 Release 5

Indexing Reference

SC19-3354-04

Notices

Before using this information and the product it supports, read the information in “Notices” on page 261.

Edition notice

This edition applies to Version 10 Release 5 of IBM® Content Manager OnDemand for Multiplatforms (program number 5724-J33) and IBM Content Manager OnDemand for z/OS® (program number 5697-CM1), and Version 7 Release 4 of IBM Content Manager OnDemand for i (program number 5770-RD1) and to all subsequent releases and modifications until otherwise indicated in new editions.

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

Contents

ibm.com® and related resources..... ix
Contacting IBM..... ix

About this publication..... xi
Who should use this publication..... xi
Accessibility information for Content Manager OnDemand..... xi

Chapter 1. Indexer overview..... 1

Chapter 2. ACIF indexer..... 3
ACIF overview..... 3
ACIF batch utility..... 5
Line data conversion to AFP™..... 5
AFP™ resources..... 6
How Content Manager OnDemand uses index information..... 7
ACIF parameters for EBCDIC data..... 8
Determining how literal values are expressed..... 10
ACIF indexer parameters..... 11
BREAKYES..... 13
CC..... 14
CCTYPE..... 14
CHARS..... 15
CONVERT..... 16
CPGID..... 17
DCFPAGENAMES..... 17
EXTENSIONS..... 18
FDEFLIB..... 21
FIELD..... 22
FILEFORMAT (Multiplatform)..... 28
FILEFORMAT (z/OS platforms)..... 29
FONTLIB..... 30
FORMDEF..... 31
FORMFEED..... 33
GROUPMAXPAGES..... 34
GROUPNAME..... 34
IMAGEOUT..... 35
INDEX..... 35
INDEXDD..... 38
INDEXOBJ..... 39
INDEXSTARTBY..... 40
INDXEXIT (Multiplatform)..... 41
INDXEXIT (z/OS platforms)..... 41
INPCCSID..... 41
INPEXIT (Multiplatform)..... 42
INPEXIT (z/OS platforms)..... 42
INPUTDD (Multiplatform)..... 42
INPUTDD (z/OS platforms)..... 43
INSERTIMM..... 43
LINECNT..... 44
LINEOFFSET..... 44

MCF2REF ... 45
MSGDD (Multiplatform) ... 46
MSGDD (z/OS platforms) ... 46
NEWPAGE ... 47
OUTCCSID ... 47
OUTEXIT (Multiplatform) ... 48
OUTEXIT (z/OS platforms) ... 48
OUTPUTDD (Multiplatform) ... 48
OUTPUTDD (z/OS platforms) ... 49
OVLYLIB (Multiplatforms) ... 49
OVLYLIB (z/OS platforms) ... 50
PAGEDEF ... 51
PARMDD (Multiplatform) ... 53
PARMDD (z/OS platforms) ... 53
PDEFLIB ... 54
PRMODE ... 55
PSEGLIB ... 56
RESEXIT ... 57
RESFILE ... 57
RESLIB ... 58
RESOBJDD (Multiplatform) ... 59
RESOBJDD (z/OS platforms) ... 59
RESTYPE ... 60
TRACE ... 62
TRACEDD (Multiplatform) ... 63
TRACEDD (z/OS platforms) ... 63
TRC ... 63
TRIGGER ... 64
UNIQUEBNGS ... 67
USERLIB ... 68
USERMASK ... 69
USERPATH ... 70
Print messages ... 71
User exits and attributes of the input file ... 71
User programming exits ... 72
Input record exit ... 72
Index record exit ... 76
Output record exit ... 78
Resource exit ... 80
User exit search order ... 83
Non-Zero return codes ... 84
Attributes of the input file ... 84
ACIF exits written in COBOL (z/OS systems) ... 85
ACIF data stream information ... 87
Tag Logical Element (TLE) structured field ... 87
Format of the resource file ... 88
ACIF processing of fully composed AFP™ files ... 89
Format of the ACIF index object file ... 90
Group-level Index Element (IEL) structured field ... 91
Page-level Index Element (IEL) structured field ... 91
Begin Document Index (BDI) structured field ... 91
Index Element (IEL) structured field ... 92
Tag Logical Element (TLE) structured field ... 92
End Document Index (EDI) structured field ... 93
Format of the ACIF output document file ... 93
Page groups ... 94
Begin Document (BDT) structured field ... 94
Begin Named Group (BNG) structured field ... 95

Tag Logical Element (TLE) structured field....95
Begin Page (BPG) structured field....95
End Named Group (ENG), End Document (EDT), and End Page (EPG) structured fields....95
Output MO:DCA data stream....96
ACIF examples....97
Example one: Bank loan report....97
Example two: Phone bill....104
Example three: Income statement....114
Example four: AFP data....127
Using ACIF in z/OS....131
Sample JCL....131
ACIF parameters....132
JCL and ACIF parameters....133
Hints and tips....135
Control statements that contain numbered lines....135
Placing TLEs in named groups....135
File transfer....136
ANSI and machine carriage controls....137
Common methods of transferring files....138
Using the Invoke Medium Map (IMM) structured field....139
Indexing considerations....139
Concatenating resources to an AFP™ file....140
Specifying the IMAGEOUT parameter....140
Running ACIF with inline resources....141
Writing inline resources to the output file....141
Using regular expressions....141

Chapter 3. 390 indexer....145
390 indexer parameters....147
AFPINDEXBUF....147
ANYEXIT....147
BREAKYES....151
CPGID....151
DJDECNT....151
DJDECOL....152
DJDETRIG....152
FIELD....152
FILEFORMAT....155
GROUPMAXPAGES....155
INDEX....156
INDEXSTARTBY....158
INDEXSTYLE....158
INDXEXIT....165
INPEXIT....167
INPEXITNEW....169
LINEOFFSET....170
MCC2ANSI....171
Triggers....172
XEROX DJDE Support....172
Using the 390 indexer....172
Content Manager OnDemand application....173
Large objects and the 390 indexer....173
The ARSLOAD program in a z/OS environment....173

Chapter 4. 400 indexer....175
400 indexer parameters....175
Unique indexing parameter reference....177

CC....177
CCTYPE....177
CONVERT....178
CPGID....178
DOCTYPE....178
FIELD....179
FILEFORMAT....185
IMAGEOUT....185
INDEX....185
INDEXOBJ....187
INDEXSTARTBY....187
INDEXSTYLE....188
STARTINDEXINGONPAGE....188
STARTTRANSACTIONFIELDSONLINE....189
STARTTRIGGERSONLINE....189
TRANSLATEPRINTCONTROL....189
TRIGGER....190
BREAK setting....193
Controlling maximum number of pages per group....194
Using Group triggers versus Float triggers....194
Using a mask when defining application fields....194
Using regular expressions....195
Regular expressions and the TRIGGER parameter....196
Regular expressions and the FIELD parameter....196
Default values for fields....196
Assigning default index values....197
Handling SCS spooled files that have AFP overlays....198
Using Tag Logical Elements (TLEs)....199
Defining multi-key indexes....199
Multi-key index example....200
Defining transaction fields....202
Transaction report example....203
Understanding Translate Print Control....205
Using system date or job run date as the value of a date field....205
Alternative for creating sample data to define triggers, indexes, and fields....207
Defining text search fields....207

Chapter 5. PDF indexer....209
How Content Manager OnDemand processes index information....211
Processing PDF input files with the graphical indexer....211
Indexing input data....214
Coordinate system....214
Indexing parameters....214
Indexing with metadata indexes....217
Indexing with internal indexes....218
Using regular expressions....220
Using a regular expression on the TRIGGER parameter....220
Using a regular expression on the FIELD parameter....221
How to create indexing parameters....221
PDF fonts and output file size....222
PDF resource collection....223
PDF indexing system requirements....223
Specifying the location of Adobe® fonts....223
PDF indexing limitations....224
Input data requirements....224
National language support for indexed PDF documents....225
PDF indexer parameters....226

BOOKMARKS....226
COORDINATES....226
DISABLECHARREORDERING....227
FIELD....227
FONTLIB....231
INDEX....232
INDEXDD....233
INDEXMODE....233
INDEXSTARTBY....234
INPUTDD....235
MSGDD....235
OUTPUTDD....236
PARMDD....236
REMOVERES....237
RESOBJDD....237
RESTYPE....238
TEMPDIR....238
TRACEDD....238
TRIGGER....239
PDF indexer messages....241
ARSPDOCI program....242
ARSPDUMP program....243
Trace facility....245

Chapter 6. Generic indexer....247
Loading data....247
Processing AFP™ data....250
Generic indexer parameters....251
CODEPAGE:....251
COMMENT:....252
GROUP_FIELD_NAME:....252
GROUP_FIELD_VALUE:....253
GROUP_FILENAME:....253
GROUP_LENGTH:....255
GROUP_OFFSET:....255
Parameter file examples....256

Chapter 7. XML indexer....259
.xsd schema file....259
Resources....260
Invocation....260

Notices....261
Trademarks....262
Terms and conditions for product documentation....263
IBM Online Privacy Statement....263

Index....265

ibm.com® and related resources

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

Support and assistance

From ibm.com, click **Support & downloads** and select the type of support that you need. From the Support Portal, you can search for product information, download fixes, open service requests, and access other tools and resources.

IBM Knowledge Center

See your online product information in IBM Knowledge Center at:

- IBM Content Manager OnDemand for Multiplatforms: https://www.ibm.com/support/knowledgecenter/SSEPCD
- IBM Content Manager OnDemand for z/OS: https://www.ibm.com/support/knowledgecenter/SSQHWE
- IBM Content Manager OnDemand for i: https://www.ibm.com/support/knowledgecenter/SSB2EG

PDF publications

See the following PDF publications for your product at:

- IBM Content Manager OnDemand for Multiplatforms: https://www.ibm.com/support/pages/node/1079037
- IBM Content Manager OnDemand for z/OS: https://www.ibm.com/support/pages/node/1079043
- IBM Content Manager OnDemand for i: http://www.ibm.com/support/docview.wss?uid=ibm10740829

Contacting IBM

For general inquiries, call 800-IBM-4YOU (800-426-4968). To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

For more information about how to contact IBM, including TTY service, see the Contact IBM website at http://www.ibm.com/contact/us/.

About this publication

This guide contains information about indexing methods, preparing index data, and using tools to index reports that you plan to store in and retrieve from IBM Content Manager OnDemand for Multiplatforms Version 10 Release 5, IBM Content Manager OnDemand for z/OS Version 10 Release 5, or IBM Content Manager OnDemand for i Version 7 Release 4. Unless otherwise specified, these products are collectively referred to in this guide as Content Manager OnDemand.

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 for administrators and other people in an organization who are responsible for preparing data to be stored in Content Manager OnDemand.

Accessibility information for Content Manager OnDemand

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

Chapter 1. Indexer overview

The indexers in IBM Content Manager OnDemand provide a way to load and store reports. Indexers determine where one document ends and the next begins, as well as which index values are to be associated with each document.

These index values are used to identify and retrieve documents for viewing or printing. Usually the index values are extracted from the content of the documents but they can also be created manually or by a custom application (an exit). Indexing may also create a resource file that contains all the resources needed to view and print a document.

The indexer is a program that provides these functions. The choice of an indexer depends on the platform, the format of the documents, and functionality needed. Content Manager OnDemand provides the following indexers:

Table 1. Indexer support by platform

Indexer	AIX®	Linux	z/OS	IBM i	Windows	Linux for System z
ACIF	✓	✓	✓		✓	✓
Generic	✓	✓	✓	✓	✓	✓
400				✓		
PDF	✓	✓		✓	✓	
XML	✓	✓	✓	✓	✓	✓
390	✓	✓	✓		✓	✓

ACIF indexer

The ACIF indexer extracts index data, and optionally converts line data to AFP and creates an AFP resource file. ACIF accepts input files that contain AFP, line data, non-formatted ASCII data, and ASCII data containing printer control characters generated on Windows or UNIX workstations. The ACIF indexer creates the index file in AFP format. The ACIF indexer is called by the ARSLOAD program during the loading process or can be run from the command line on Content Manager OnDemand for Multiplatforms or as a batch job on z/OS . You can use the OnDemand Administrator graphical indexer to create indexing parameters for the ACIF indexer.

390 indexer

The 390 indexer extracts indexes and loads documents from line data, AFP, and DJDE reports. Other data types, such as PDF and TIFF images, can be captured by using the ANYSTORE exit. This provides a method to capture documents of any type and size (including those greater than 2 GB) into Content Manager OnDemand. The 390 indexer loads the index values and documents directly into the Content Manager OnDemand database in a single pass, without needing to create intermediary files.

400 indexer

The 400 indexer extracts index data and creates an AFP resource file. The 400 indexer accepts SCS, SCS-Extended, Advanced Function Presentation (AFP), and Line spooled files. The 400 indexer creates the index file in the Generic index format (see Generic indexer). The 400 indexer is called by the **ADDRPTOND** command for SCS, SCS-Extended, AFP, and Line spooled files. It is the primary indexer used when you are running on an IBM i system and will be used by default for SCS, SCS-Extended, AFP, and Line spooled files. You can use the OnDemand Administrator graphical indexer to create indexing parameters for the 400 indexer.

PDF indexer

The PDF indexer extracts index data and creates a PDF resource file from Adobe PDF files. The PDF indexer creates the index file in the Generic index format (see Generic indexer). Although the PDF indexer is included with Content Manager OnDemand for Multiplatforms, you must purchase the PDF

indexer before you are authorized to use it. If you are using Content Manager OnDemand for i, the PDF indexer must be purchased separately. The PDF indexer is called by the ARSLOAD program during the loading process or can be run from the command line on Multiplatforms. On IBM i operating environments, the PDF indexer can be called by the **ADDRPTOND** command or by the ARSLOAD program. You can use the OnDemand Administrator graphical indexer to create indexing parameters for the PDF indexer.

Restriction: The PDF indexer is not supported on z/OS or Linux for System z platforms.

Generic indexer (generic index format)
The Generic indexer refers to a specific file format, referred to as the generic index format. You can use this file format to load index data for any type of input file that needs to be stored in Content Manager OnDemand. The Generic indexer allows administrators to specify indexing information for input data that you cannot or do not want to index with the other indexers. For example, suppose that you want to load files into the system that were created by using a word processor. By creating a generic index file, the files can be stored in the system in the same format in which they were created. The Generic indexer is called by the ARSLOAD program during the loading process. You cannot use the OnDemand Administrator graphical indexer to create indexing parameters for the Generic indexer. A generic index file is usually created manually or by an external application.

XML indexer
The XML indexer extracts index data from XML files. It creates the index file in the generic index format. The XML indexer is called by the ARSLOAD program during the loading process.

Chapter 2. ACIF indexer

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.

ACIF overview

ACIF is a batch utility that provides indexing functions, print data stream conversions, and AFP resource collection.

ACIF is a powerful tool for indexing the print data streams of ACIF application programs, EBCDIC data on z/OS , unformatted ASCII data, and ASCII data containing printer control characters that is generated on multiplatform 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 a Content Manager 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 Content Manager 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. During document retrievalContent Manager OnDemand stores the resources in cache storage or archive media 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 Content Manager 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 or TYPE=PAGERANGE option. For more information, see the discussion of TYPE=PAGE in "INDEX" on page 35.

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 a Content Manager OnDemand library or object server or on System z on which the ACIF programs are installed.

- When you run ACIF on a 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 guide 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 System z requires:
  - Print Services Facility (PSF) for OS/390® Version 4 Release 4 or later.
  - Content Manager OnDemand version of ACIF, which can be ordered without charge by customers who are entitled to Content Manager OnDemand.

Indexing

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. 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.

ACIF batch utility

ACIF is a batch utility that provides these main functions.

Sophisticated indexing functions
ACIF can logically divide reports into individual items, such as statements, policies, and bills. You can define up to 128 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 Content Manager 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.

Line data conversion 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 data
- Line data
- Mixed-mode data
- Unformatted ASCII

AFP™ data
The AFP data stream is a superset of the MO:DCA data stream.

The AFP data stream 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.

Mixed Object Document Content Architecture™ Data
ACIF supports MO:DCA data as a valid input data stream.

The following restrictions apply:

- 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 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 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 data may not contain the same structured fields or the structured fields may not appear in the same order.

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.

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.

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.

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 (or libraries in z/OS). The directories must reside 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. Content Manager OnDemand loads the resource file at the same time it loads the indexed report files.

When you store a report in Content Manager 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. The following table lists typical values for the RESTYPE parameter.

Table 2. Collecting Resources

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

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

- Identifies the resources requested by the print file.

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

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

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

- Includes the name of the output document in the resource file and the name of the resource file in the output document.

  This provides a method of correlating resources files with the appropriate output document.
- If a resource is inline and ACIF is collecting that type of resource, the resource will be saved in the resource file regardless of whether it is used in the document, unless `EXTENSIONS=RESORDER` is specified in the ACIF parameters.

  Another method to remove unwanted resources from the resource file is to use a resource exit.

How Content Manager OnDemand uses index information

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.

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. Figure 1 on page 7 shows an overview of the index creation and loading process.

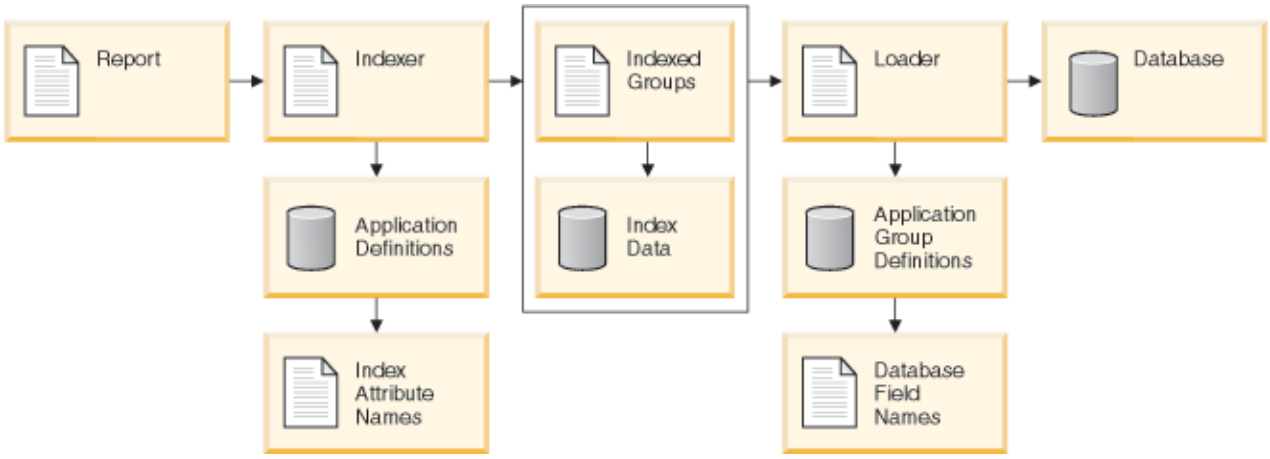


Figure 1. Indexing reports

You typically create a 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. Content Manager 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 Content Manager 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. Content Manager OnDemand searches the database for documents that contain index values (ACIF attribute values) that match the search values entered by the user. Content Manager OnDemand lists the documents that match the query. When the user selects a document for viewing, the

Content Manager 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 or TYPE=PAGERANGE option. For more information, see the discussion of TYPE=PAGE in “INDEX” on page 35.

ACIF parameters for EBCDIC data

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

Accessing reports

Reports generated on a z/OS system are typically transmitted to a 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.

Important: 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 Content Manager OnDemand to index and subsequently read the file.

Creating indexing parameters

If you index reports on a 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.

About this task

If you index reports on a z/OS system, you 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 */
```

Specifying indexing parameters

To locate indexing attributes in the sample report, two TRIGGER parameters are required that tell ACIF to examine the first byte of an input record and then look for a specific hexadecimal string.

Before you begin

The sample report uses the following data values for indexing attributes:

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

About this task

Important: If you are processing EBCDIC data on z/OS , the trigger and index values can be expressed as regular character strings (or hex), but if you are processing EBCDIC data on multiplatform operating systems, then they must be expressed in hexadecimal.

To locate these indexing attributes in the sample report, two TRIGGER parameters are required. The first trigger tells ACIF to examine the first byte of every input record until it finds the occurrence of an ANSI skip-to-channel one carriage control character.

After locating a record containing a character `'1'` (on z/OS systems) or hexadecimal `X'F1'` (on multiplatform systems) in the first byte, ACIF uses the second trigger to look for the following string starting in column 72 of the same input record:

- On z/OS systems: the character string `'Page 0001'`
- On multiplatform systems: the hexadecimal string `X'D7C1C7C540F0F0F0F1'` (PAGE 0001)

When this condition is found, a new statement exists, and the record containing a character `'1'` (on z/OS systems) or hexadecimal `X'F1'` (on multiplatform systems) in the first byte is considered the anchor record. ACIF uses the anchor record to locate index values. The trigger specifications are expressed as follows:

Indexing EBCDIC data (part 1 of 4) on z/OS (using regular character strings):

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

Indexing EBCDIC data (part 1 of 4) on multiplatform systems (in hexadecimal format):

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

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

To create the indexing tag for the customer name attribute, define the following string as the indexing attribute:

- On z/OS systems: the character string `'custnam'`
- On multiplatform systems: the hexadecimal string `X'839AA2A3958194'` (custnam)

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:

Indexing EBCDIC data (part 2 of 4) on z/OS (using regular character strings):

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

Indexing EBCDIC data (part 2 of 4) on multiplatform systems (in hexadecimal format):

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

To create the indexing tag for the statement date attribute, define the following string as the indexing attribute:

- On z/OS systems: the character string `'sdate'`
- On multiplatform systems: the hexadecimal string `X'A28481A385'` (sdate)

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:

Indexing EBCDIC data (part 3 of 4) on z/OS (using regular character strings):

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

Indexing EBCDIC data (part 3 of 4) on multiplatform systems (in hexadecimal format):

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

To create the indexing tag for the account number attribute, define the following string as the indexing attribute:

- On z/OS systems: the character string `'acctnum'`
- On multiplatform systems: the hexadecimal string `X'818383A395A494'` (acctnum)

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:

Indexing EBCDIC data (part 4 of 4) on z/OS (using regular character strings):

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

Indexing EBCDIC data (part 4 of 4) on multiplatform systems (in hexadecimal format):

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

After indexing the report, Content Manager OnDemand stores the index values in the database for each of the three indexing attributes for each statement in the input data stream. Using a Content Manager OnDemand client program, you 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 can be expressed in character data strings. The following example, 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. The following is an example of a UNIX or Windows parameter file for ASCII input data:

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

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. The following example, 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. The following is an example of a UNIX or Windows parameter file for EBCDIC input data:

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

ACIF indexer parameters

Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

This parameter reference 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 ACIF is specified as the indexer in the application. The ARSLOAD program ignores the **INDEXDD**, **INPUTDD**, **MSGDD**, **OUTPUTDD**, **PARMDD**, and **RESOBJDD** parameters, if specified. If you run ACIF from the command prompt, you must specify the values of the **INDEXDD**, **INPUTDD** , **MSGDD**, **OUTPUTDD**, **PARMDD**, and **RESOBJDD** parameters.

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, unless you specify a regular expression.
- A maximum of 16 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 16 TRIGGER parameters.
- A maximum of 128 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 Content Manager 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. Content Manager 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 or TYPE=PAGERANGE option. For more information, see the discussion of TYPE=PAGE in "INDEX" on page 35.
- You can concatenate field parameters to form an index.
- A maximum of 128 index parameters can be specified.
- The default behavior of ACIF is to create a new group and extract new index values when one or more of the fixed group index values change or the GROUPMAXPAGES value is reached. For information on changing the default behavior, see "BREAKYES" on page 13.

The following illustration shows 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 2. Indexing a report

The following indexing parameters could be used to generate index data for the report shown in the illustration. 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.

Related reference

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.

TRIGGER
Identifies locations and string values required to uniquely identify the beginning of a group and the locations and string values of fields used to define indexes. You must define at least one trigger and can define up to 16 triggers.

TRIGGER
Identifies locations and string values required to uniquely identify the beginning of a group and the locations and string values of fields used to define indexes. You must define at least one trigger and can define up to 16 triggers.

USERMASK
Identifies a symbol and string used to match a field.

INDEXDD
Determines the name or the full path name of the index object file, where ACIF writes indexing information.

INPUTDD (Multiplatform)
Identifies the file name or full path name of the input file that ACIF will process.

MSGDD (Multiplatform)
Determines the name or the full path name of the file where ACIF writes error messages.

OUTPUTDD (Multiplatform)
Identifies the name or the full path name of the output file.

PARMDD (Multiplatform)
Identifies the name or the full path name of the file that contains the ACIF parameters, options, and data values.

RESOBJDD (Multiplatform)
Identifies the name or the full path name of the resource file produced by ACIF.

BREAKYES

Determines how ACIF starts a new document.

Required
No

Default Value
OR

Data Type
AFP, Line

Syntax

BREAKYES=*value*

Options and values

The *value* parameter can be:

AND
The multiple index parameters with BREAK=YES are all "AND'ed" together. In other words, all the index values must change for the current page to be considered the start of a new document.

OR
The multiple index parameters with BREAK=YES are all "OR'ed" together. In other words, a change to any index value causes the current page to be considered the start of a new document.

For the ACIF indexer or the 400 indexer, the default value for BREAKYES is OR. If a value other than AND or OR is specified for this parameter, the indexer stops processing and issues an error message.

For the 390 indexer, the default value for BREAKYES is AND. If a value other than AND or OR is specified for this parameter, the indexer issues a warning message and continues processing as if AND were specified.

CC

Determines whether the input contains carriage-control characters.

Required
No

Default Value
YES

Data Type
AFP, Line

Syntax

CC=*value*

Options and values

The *value* parameter can be set to YES or NO:

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” on page 14

CCTYPE

If the data contains carriage control characters, determines the type of carriage-control characters.

Required
No

Default Value
A (on a z/OS system) or Z (on all other platforms)

Data Type
AFP, Line

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* parameter can be Z, A, or M:

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" on page 14

CHARS

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.

Required
No

Default Value
(None)

Data Type
AFP

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.

Use the CHARS parameter to specify coded fonts in a font library having names of six or fewer characters (including the prefix). You can rename any fonts having more than six characters or use a text editor to create new coded fonts for use with the CHARS parameter. When ACIF is used to convert line data or mixed-mode data, you must specify a page definition with the PAGEDEF parameter. You can then specify the fonts either in the page definition or with the CHARS parameter, but not both. You cannot mix fonts specified in a page definition with fonts specified with CHARS for a single file. If you use the CHARS parameter to specify fonts, but you also use the PAGEDEF parameter to specify a page definition that names fonts, the CHARS parameter is ignored. Therefore, if your page definition names fonts, you should not use the CHARS parameter.

Syntax

CHARS=*fontlist*

Options and values

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

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

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

The fonts that you specify must reside in a library that is specified with **FONTLIB**, **USERLIB**, or **RESLIB** (UNIX or Windows servers) parameters.

Related parameters

- "FONTLIB" on page 30
- "PAGEDEF" on page 51
- "USERLIB" on page 68

CONVERT

Determines whether ACIF converts the input data to AFP.

Required
No

Default Value
YES

Data Type
AFP, Line

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.

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 Content Manager 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 if it is specified, it must be set to the value YES.

NO
ACIF does not convert the input data to AFP.

Related parameters

"RESTYPE" on page 60

CPGID

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

Required
No

Default Value

- Multiplatform systems: 850 (ASCII)
- z/OS systems: 500 (EBCDIC)

Data Type
AFP, Line

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

The code page identifier is used by the Content Manager OnDemand client programs to display indexing information. The client programs use this identifier with code page translation tables to represent the index attribute and value data. If a non-decimal value is specified, ACIF reports an error condition and ends processing.

On z/OS systems, for code-page numbers less than 100, add leading zeros (for example, 037).

Syntax

CPGID=*value*

Options and values

The *value* can be:

850 (ASCII) 500 (EBCDIC)
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

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

Required
No

Default Value
NO

Data Type
AFP, Line

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

Determines the extended options that ACIF uses.

Required
No

Default Value
NONE

Data Type
AFP, Line

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.

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

ADDTLE
If the input file is fully composed AFP, ACIF will add group TLEs to the output and index files. The TLE attribute name is specified as an **INDEX** parameter, and the TLE attribute value is specified as a **FIELD** parameter containing a constant value. For example, given the following definitions ACIF will add two TLEs: the first will have an attribute name of "DEPT" and a value of "111", the second will have an attribute name of "CODE" and a value of "R7". The TLEs will be added to each group. Note that the index name and value must be encoded in the correct code page.

```
CC=YES
CCTYPE=A
CPGID=500
FORMDEF=F1A10110
EXTENSIONS=ADDTLE
RESTYPE=FDEF,PSEG,OVLY
FIELD1=X'F1F1F1'
INDEX1=X'C4C5D7E3',FIELD1
FIELD2=X'D9F7'
INDEX2=X'C3D6C4C5',FIELD2
```

Usage notes:

1. Input must be AFP that contains BNG/ENG pairs.
2. No **TRIGGER** parameters can be specified with **ADDTLE**; the **FIELD** parameters must contain constant values.

3. By default ACIF will use the value of *INDEX1* for the **GROUPNAME**. *INDEX1* must be defined or the **GROUPNAME** parameter must be specified to indicate which of the constant fields ACIF should use for the **GROUPNAME**, for example:

```
CC=YES
CCTYPE=A
CPGID=500
FORMDEF=F1A10110
EXTENSIONS=ADDTLE
RESTYPE=FDEF,PSEG,OVLY
FIELD3=X'F1F1F1'
INDEX3=X'C4C5D7E3',FIELD3
GROUPNAME=INDEX3
```

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 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.

If no indexing information is found, the file will 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. 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

When you specify this parameter, note the following restrictions:

- 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.
- 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.
- Ensure that you express the trigger and index names in Big Endian.
- Ensure that you extract the field values from the document in Big Endian format.
- The **IDXCPGID** parameter can be used with both **CONVERT**=YES and **CONVERT**=NO.
- Do not use a mask on the **FIELD** parameter when you use the **IDXCPGID** parameter.
- Do not use the **IDXCPGID** parameter if the input is AFP or mixed-mode.

- ACIF issues an error message if the **IDXCPGID** parameter is specified with the **PASSPF** parameter. If **EXTENSIONS**=ALL is specified, **PASSPF** is ignored and the **IDXCPGID** parameter is used.

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 example illustrates these 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 */
  ```

PASSPF

Specifies that ACIF should pass the Begin Print File (BPF) and End Print File (EPF) structured fields, which define the boundaries of the print data, to the output file when they are found in the input file. If this value is not specified, ACIF discards the BPF/EPF pair. This parameter also verifies whether a BPF/EPF structured field pair that the input record exit tried to insert is actually inserted. If this value is not specified, and the input record tries to insert a BPF/EPF pair, the attempt fails and the pair is discarded.

Note:

1. Be careful using PASSPF. If the output file contains BPF and EPF structured fields and it is concatenated with the resource file, the resulting MO:DCA-P data stream is not valid.
2. This value is not used when the input file is line data because line data does not contain BPF and EPF structured fields.
3. When PASSPF is specified, ACIF passes all Begin Document (BDT) and End Document (EDT) structured field pairs from the MO:DCA-P input file to the output data stream without adding the normal comment and timestamp triplets.
4. ACIF issues an error message if PASSPF is specified with the IDXCPGID parameter. If EXTENSIONS=ALL is specified, PASSPF is ignored and IDXCPGID is used.
5. ACIF does not verify whether the input file is MO:DCA IS/3 compliant.

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.

If there are many inline resources and little internal memory available, the system might 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

- "CPGID" on page 17
- "IMAGEOUT" on page 35

FDEFLIB

On multiplatform systems, **FDEFLIB** identifies directories in which form definitions are stored. On z/OS systems, **FDEFLIB** specifies the data sets that compose the form definition library.

Required
No

Default Value
(None)

Data Type
AFP

FDEFLIB in a multiplatform environment

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 UNIX servers, the directory `/usr/lpp/psf/reslib`, if it exists.

FDEFLIB in a z/OS environment

You can specify a maximum of 16 data sets. For example:

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

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

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

- Data sets must be specified as fully-qualified names without quotation marks.
- Separate data set names with a comma.

- For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.
- The FDEFLIB parameter is required if the USERLIB parameter is not specified. If the FDEFLIB parameter is not specified, ACIF reports an error condition and ends processing.

Syntax and options in a multiplatform environment

FDEFLIB=*pathlist*

The value to provide for the **FDEFLIB** is a valid path name.

The *pathlist* is a string of one or more valid path names, for example: `FDEFLIB=/tmp:/usr/resources:/opt/IBM/ondemand/V10.5/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.

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

Syntax and options in a z/OS environment

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

You can specify a maximum of 16 data sets. For example:

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

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

Related parameters

- “RESLIB” on page 58
- “USERLIB” on page 68

FIELD

Identifies the location of index data and can provide default and constant index values. You must define at least one field.

Required
Yes

Default Value
(None)

Data Type
AFP, Line

You can define up to 128 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

To specify a trigger field, you must assign the field a number prefixed by **FIELD**, then specify the location of the field on the report.

Multiplatform syntax:

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

z/OS syntax:

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

The **REGEX** parameter is not available on z/OS operating systems.

Trigger field options and values

There are several different values to specify for the syntax of a trigger field.

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.

When you specify 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 is within the record length. In this case, ACIF adds padding blanks to complete 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.

Use **BASE**=0 if the field data always starts in a specific column. Use **BASE**=**TRIGGER** if the field data does not always start in a specific column, but is always offset from the trigger string value a specific number of columns.

For example, a trigger occurs in the second record on a page. The trigger string value can begin in any column in the record. A field based on this trigger occurs in the trigger record. The starting column number of the field is always ten bytes from the starting column number of the trigger. Specify **BASE**=**TRIGGER** and a column offset of ten so that ACIF correctly locates the field, regardless of the starting column of the trigger string value.

MASK='@#=¬^%'

Specifies a pattern of symbols that ACIF uses to match data located in the field columns. If the data matches the MASK, then ACIF selects the field. Important: If you specify a MASK, then the field must

be based on a floating trigger. An INDEX parameter that is based on the field cannot include any other fields and must not create grouprange or pagerange indexes.

You can specify the following symbols in the MASK:

@
Matches alphabetic characters.

#
Matches numeric characters.

=
Matches any character.

¬
Matches any non-blank character.

^
Matches any non-blank character.

%
Matches the blank character and numeric characters.

For example, given the following definitions:

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

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

REGEX='regular expression'
ACIF extracts the text specified by the column and length values. After the field is extracted, ACIF applies the regular expression to the text. Any text that matches the regular expression is extracted for the field. If the matching text is shorter than the length specified in the **FIELD**, it is padded with blanks until it equals the length. If the regular expression does not match any text in the field, the following occurs:

For a field based on a Group trigger the default value specified on the **FIELD** is used. If no default value is specified, ACIF ends with error message APK488.

For a field based on a Float trigger, there is no error and the default value specified on the **FIELD** is not used. In this case the load process will use the default value specified in the Application.

If the record is only long enough to contain part of the field, the regular expression is applied only to the portion of the record that is present.

The regular expression must be specified in the code page given by the CPGID parameter. It can be specified in hexadecimal.

The maximum length of the regular expression is 250 bytes. A mask and a regular expression cannot both be specified on the same FIELD parameter.

Restriction: The **REGEX** parameter is not available on z/OS operating systems.

DEFAULT='*value*'
Determines the default value for the index when a record is not long enough to contain the field data, or (on multiplatform systems only) if a regular expression does not match any 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'. Given the following definition:

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

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

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

Trigger field examples

One example shows how to specify the location of the field that begins at a specific column, a second example shows how to specify the location of the field as an offset.

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, for example:

```
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)
```

On multiplatform systems, the following field parameter causes ACIF to apply the regular expression to columns 13 through 30 of the record that contains the trigger string value. Any text that matches will be extracted for the field value. This regular expression is designed to extract dates of the form “January 20, 1970”.

```
TRIGGER1=*,1,'1'
FIELD1=0,13,18,(REGEX='[A-Z][a-z]+ [0-9]+, [0-9]{4}')
```

Constant field syntax

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.

FIELD*n=constant*

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. The trigger field cannot be based on a floating trigger.

Constant field 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 by 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. The constant value can be 250 bytes of character data, or 125 hexadecimal characters, since each hexadecimal character contains 2 bytes. ACIF does not validate the actual content of the supplied data.

Constant field examples

One example shows how to store the same string of hexadecimal characters in each INDEX3 value it creates, the second example shows how to concatenate a constant value with the index value extracted from the data.

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

```
FIELD3=X'F0F0F0F0F0F0F0F0F0'
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 by 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

To specify a transaction field, the syntax must include a field name and the location on the report to obtain the field value.

Multiplatform syntax:

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

z/OS syntax:

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

The **REGEX** parameter is not available on z/OS operating systems.

Transaction field 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 or REGEX keyword. The implied length of an OFFSET value 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. You can specify either a mask or a regular expression, but not both. When you specify the MASK keyword, you must also specify the OFFSET keyword. When you define a transaction field that includes a mask, an

INDEX parameter based on the field cannot reference any other fields. An **INDEX** parameter based on a transaction field that includes a mask must create grouprange or pagegrange indexes. Valid mask symbols include:

*****
Not literal; matches a user-defined mask.

@
Matches alphabetic characters.

#
Matches numeric characters.

¬
Matches any nonblank character.

^
Matches any nonblank 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 cause ACIF to search for a hexadecimal C1 followed by four numeric characters (hexadecimal F0-F9), a hexadecimal 60, and two numeric characters (hexadecimal F0-F9):

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

REGEX='regular expression'
The regular expression that ACIF matches with data located in the field columns. The regular expression must be specified in the code page given by the CPGID parameter, and can be from 1 to 250 bytes in length. The regular expression can be specified in hexadecimal. You can specify either a mask or a regular expression, but not both.

When you specify the REGEX keyword, you must also specify the OFFSET keyword. When you define a transaction field that includes a regular expression, an **INDEX** parameter based on the field cannot reference any other fields. An **INDEX** parameter based on a transaction field that includes a regular expression must create grouprange or pagegrange indexes. Here are some examples of common regular expressions:

Table 3. Common regular expressions

Regular expression	Results
Account	Finds the characters "Account." By default searches are case sensitive.
[A-Z]	Finds one uppercase letter.
[A-Z]{3}	Finds three consecutive uppercase letters.
[0-9]{5}	Finds five consecutive digits.
[0-9]+	Finds one or more digits.
[^a-z]	Finds everything except lowercase a to z.
\s	Finds one whitespace character (space, tab, etc).
\S	Finds any character except for whitespace.

For example, the following definitions:

```
CPGID=850
FIELD3=*,*,8,(OFFSET=(10:17),REGEX='A[0-9]{4}-[0-9]{2}',ORDER=BYROW)
```

Cause ACIF to search columns ten through seventeen for a hexadecimal 41 followed by four numeric characters (hexadecimal 30-39), a hexadecimal 2D, and two numeric characters (hexadecimal 30-39). The match must begin in the first column specified by the **OFFSET** parameter.

Restriction: The **REGEX** parameter is not available on z/OS operating systems.

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 can 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
```

Transaction field examples

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

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

The following field parameter causes ACIF to locate three digits followed by a dash, followed by four digits, that begin in column 59 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.

```
FIELD3=*,*,12,(OFFSET=(59:70),ORDER=BYROW, REGEX='[0-9]{3}-[0-9]{4}')
```

Related parameters

- "CPGID" on page 17
- "INDEX" on page 35
- "TRIGGER" on page 64

FILEFORMAT (Multiplatform)

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

Required
No

Default Value
STREAM

Data Type
AFP, Line

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 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. The length does not include the length of the two-byte prefix. 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.

FILEFORMAT (z/OS platforms)

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

Required
No

Default Value
NONE

Data Type
Line

Syntax
FILEFORMAT=*value*

Options and values

The value can be HFSOUT. For example:

```
FILEFORMAT=HFSOUT
```

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

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

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

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

FONTLIB

On multiplatform systems, **FONTLIB** identifies the directories in which fonts are stored. On z/OS , **FONTLIB** specifies the data sets that compose the font library.

Required
No

Default Value
(None)

Data Type
AFP

FONTLIB in a multiplatform environment

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 UNIX servers, the directory `/usr/lpp/psf/reslib`, if it exists.
6. On UNIX servers, the directory `/usr/lpp/ipfonts`, if it exists.
7. On UNIX servers, the directory `/usr/lpp/afpfonts`, if it exists.
8. On UNIX servers, the directory `/usr/lpp/psf/fontlib`, if it exists.

FONTLIB in a z/OS environment

You can specify a maximum of 16 data sets. For example:

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

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

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

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

Syntax and options in a multiplatform environment

FONTLIB=*pathlist*

The value to provide for the **FONTLIB** parameter is a string of one or more valid path names.

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

```
FONTLIB=/tmp:/usr/resources:/opt/IBM/ondemand/V10.5/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.

Syntax and options in a z/OS environment

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

You can specify a maximum of 16 data sets. For example:

```
FONTLIB=SYS1.FDEFLIB,USER.PDEFLIB
```

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

Related parameters

- "RESLIB" on page 58
- "USERLIB" on page 68

FORMDEF

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

Required
Yes

Default Value
(None)

Data Type
AFP

The form definition can be located:

- inline within the input file or data set.
- in a user library referenced in the **USERLIB** parameter
- in a library referenced in the **FDEFLIB** parameter.
- in a library referenced in the **RESLIB** parameter (Multiplatform only).

To use an inline form definition:

1. Include an inline form definition in the input file or data set.
2. Specify **CC**=YES to indicate that the input file or data set contains carriage control characters. If the length of the records in the form definition is less than or equal to the logical record length defined for the input file or data set, you can specify fixed length records for the record format:
   - On UNIX and Windows systems:

     ```
     FILEFORMAT=RECORD,n
     ```

     (where n is the logical record length defined for the input file).
   - On z/OS systems, specify record format FBA (fixed block with ANSI carriage control characters) or FBM (fixed block with machine carriage control characters) for the input data set.

If the length of the records in the form definition is greater than the logical record length defined for the input file or data set, you must specify variable length records:

- On UNIX and Windows systems:

  ```
  FILEFORMAT=RECORD
  ```

  The first two bytes of each record determine the record length.
- On z/OS systems, specify record format VBA (variable blocked with ANSI carriage control characters) or VBM (variable blocked with machine carriage control characters) for the input data set.

3. Specify the **FORMDEF** parameter with one of these values:

   - `fdefname`, which is the name of the inline form definition.

     If the name specified in the **FORMDEF** parameter does not match the name of an inline form definition, ACIF looks for the form definition in the **FORMDEF** resource library search path.

     Note: On UNIX servers, the `fdefname` is case sensitive.
   - DUMMY

     If you specify **FORMDEF**=DUMMY but the file does not include an inline form definition, ACIF looks for the form definition named DUMMY. If ACIF cannot find a form definition named DUMMY, it reports an error and ends processing.

     Note: On z/OS servers, DUMMY must be specified in all uppercase letters.

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

On multiplatform systems:

```
FORMDEF=MEMO
FDEFLIB=/resources
```

On z/OS systems:

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

Syntax

FORMDEF=*fdefname*

Options and values

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

Notes

If the file name of the form definition includes a file extension, do not use the file extension when specifying the form definition. However, the file type must be FDEF3820, FDEF38PP, or FDE (or no file type).

For example, to use a form definition named `MEMO.FDEF38PP`, specify **FORMDEF**=`MEMO`.

If CONVERT=YES is specified, ACIF requires a form definition to process the input file (even though the form definition actually gets used at print time). If you do not specify the **FORMDEF** parameter or you specify **FORMDEF** without a form definition file name, ACIF reports an error condition and ends processing.

Related parameters

"FDEFLIB" on page 21

FORMFEED

Identifies whether the input file contains form feed characters (x'0C') to indicate a new page.

Required
No

Default Value
No

Data Type
Line

Attention: The **FORMFEED** parameter is not available with the 400 indexer on IBM i.

ACIF will recognize the form feed character as the start of a new page. Set this parameter in order to process the file without ACIF changing the data. If you do not use the **FORMFEED** parameter, the ACIF input exit `asciinp` is required in order to load files that contain the form feed character. In Content Manager OnDemand `asciinp` removes the x'0C' and inserts an ANSI carriage control at the beginning of each line.

Attention: 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.

Considerations:

1. **FORMFEED**=*YES* is only allowed with the **CONVERT**=*NO*, **CC**=*NO*, and **TRC**=*NO* parameters. If you specify **CONVERT**=*YES* (the default), **CC**=*YES* (the default), or **TRC**=*YES* and **FORMFEED**=*YES*, ACIF will issue error message `APK493S:APK493S A VALUE OF YES FOR THE FORMFEED PARAMETER IS NOT ALLOWED WITH THE CONVERT=YES, CC=YES, OR TRC=YES PARAMETERS.`
2. If you wish to convert an input file that contains **FORMFEED** characters to AFP, use the `asciinp` exit along with **CC**=*YES*, **CCTYPE**=*A*, and **CONVERT**=*YES*.
3. If you have been using the `asciinp` exit and wish to use **FORMFEED**=*YES* instead, you must create a new Application, because the Application and ACIF parameters used with the `asciinp` exit are incompatible with the parameters used for the **FORMFEED** parameter.
4. The **NEWPAGE** parameter is only in effect when **CC**=*YES*.

Syntax

FORMFEED=*value*

Options and values

The *value* can be set to YES or NO:

YES
ACIF will recognize the form feed character (x'0C') as the start of a new page.

NO
ACIF will not recognize the form feed character (x'0C') as the start of a new page.

Related parameters

- “CC” on page 14
- “CCTYPE” on page 14
- “NEWPAGE” on page 47

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 to 9999.

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

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

Syntax

GROUPMAXPAGES=*value*

Options and values

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

Related parameters
"INDEX" on page 35

GROUPNAME

Determines which of the 128 possible index values should be used as the group name for each index group.

Required
No

Default Value
INDEX1

Data Type
AFP

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. Content Manager 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" on page 35

IMAGEOUT

Determines the format of the image data produced by ACIF.

Required
No

Default Value
IOCA

Data Type
AFP, Line

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" on page 18

INDEX

Identifies the index name, the field or fields on which the index is based, and the type of index ACIF generates.

Required
Yes

Default Value
(None)

Data Type
AFP, Line

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 128 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

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

Options and values

- *n*

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

- *name*

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

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

- **FIELD**nn

  The name of the field parameter or parameters ACIF uses to locate the index. You can specify a maximum of 128 field parameters. 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 or the regular expression 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.

  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.

  This type of page-level information is generated by specifying the INDEXOBJ=ALL parameter, 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 or the regular expression 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.

  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.

  This type of page-level information is generated by specifying the INDEXOBJ=ALL parameter, 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.

Group index example

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=x'998481A385',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=x'95819485',FIELD1,(TYPE=GROUP,BREAK=YES)
INDEX2=x'818383A46D95A494',FIELD2,(TYPE=GROUP,BREAK=NO)
```

Grouprange index example
The Grouprange 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=x'939681956D95A494',FIELD2,(TYPE=GROUPRANGE,BREAK=NO)
GROUPMAXPAGES=100
```

Page index example
The Page 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=x'E2A482A396A38193',FIELD3,(TYPE=PAGE,BREAK=NO)
```

Related parameters

- “FIELD” on page 22
- “INDEXOBJ” on page 39

Related concepts
Key concepts

INDEXDD

Determines the name or the full path name of the index object file, where ACIF writes indexing information.

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

Required
No

Default Value
INDEX

Data Type
AFP, Line

When ACIF is indexing the input file, it writes indexing information to the specified DD name. 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.

On z/OS systems, the suggested DCB characteristics for the file are:

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

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

- Physical sequential format

Syntax and options on multiplatform systems

INDEXDD=*filename*

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

Syntax and options on z/OS systems

INDEXDD=*DD name*

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

Related concepts

ACIF indexer parameters

Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

INDEXOBJ

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

Required
No

Default Value
GROUP

Data Type
AFP, Line

Syntax
INDEXOBJ=*value*

Options and values

The *value* can be:

- **GROUP**

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

  Note: If you define page-level indexes and specify INDEXOBJ=GROUP, 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 Content Manager 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 35.

- **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 Content Manager 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 its own. This file is suitable for printing, but should not be used with indexing because the resultant index will not be MO:DCA compliant and may not be processed correctly by programs which use the index, such as Content Manager OnDemand.

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

```
INDEXOBJ=BDTLY
```

Related parameters
"INDEX" on page 35

INDEXSTARTBY

Determines the page number by which ACIF must find a group indexing field.

Required
No

Default Value
1

Data Type
AFP, Line

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.

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.

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.

INDXEXIT (Multiplatform)

Identifies the name or the full path name of the index record exit program.

Required
No

Default Value
(None)

Data Type
AFP, Line

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, see the discussion in "User exits and attributes of the input file" on page 71.

Syntax
INDXEXIT=*name*

Options and values

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

Related concepts
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.

INDXEXIT (z/OS platforms)

Specifies the name of the module ACIF loads during initialization and subsequently calls for every record (structured field) it writes to the index object file (specified with the INDEXDD parameter).

Required
No

Default Value
(None)

Data Type
AFP, Line

If this parameter is not specified, no index record exit is used.

Syntax
INDXEXIT=*modulename*

Options and values

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

INPCCSID

Specifies a valid coded character set identifier (CCSID) for the input code page you want to convert to another CCSID. This parameter can be used by an input record exit program, such as apka2e or asciinpe to translate input data streams.

Syntax

INPCCSID=ccsid

Options and values

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

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

INPEXIT (Multiplatform)

Identifies the name or the full path name of the input record exit program.

Required
No

Default Value
(None)

Data Type
AFP, Line

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, see the discussion in "User exits and attributes of the input file" on page 71.

Syntax
INPEXIT=*name*

Options and values

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

INPEXIT (z/OS platforms)

Specifies the name of the module ACIF loads during initialization and subsequently calls for every input record it reads from the input file (specified with the INPUTDD parameter).

Required
No

Default Value
(None)

Data Type
AFP, Line

If this parameter is not specified, no input record exit is used.

Syntax
INPEXIT=*modulename*

Options and values

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

INPUTDD (Multiplatform)

Identifies the file name or full path name of the input file that ACIF will process.

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

Required
No

Default Value
stdin

Data Type
AFP, Line

If you do not specify the INPUTDD parameter, ACIF uses standard input.

Syntax
INPUTDD=*filename*

Options and values

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

Related concepts
ACIF indexer parameters
Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

INPUTDD (z/OS platforms)

Specifies the DD name for the input file ACIF processes.

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

Required
No

Default Value
INPUT

Data Type
AFP, Line

Syntax
INPUTDD=*DD name*

Options and values

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

INSERTIMM

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

Required
No

Default Value
NO

Data Type
AFP

Syntax

INSERTIMM=*value*

Options and values

The *value* can be:

NO
ACIF does not insert IMMs into the output data.

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

Important: The INSERTIMM parameter should be used carefully. 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

"FORMDEF" on page 31

LINECNT

For unconverted line data, determines the maximum number of lines per page. This parameter tells ACIF when to create page breaks.

Required
No

Default Value
0

Data Type
Line

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

- "CC" on page 14
- "CONVERT" on page 16

LINEOFFSET

Determines whether ANSI carriage-control characters are used to calculate the record offsets when determining the location of the fields.

Required
No

Default Value
ASREAD

Data Type
AFP, Line

Restriction: The **LINEOFFSET** parameter is not available in IBM Content Manager OnDemand for z/OS.

Only the 0 (space two lines) and the dash (space three lines) are supported. The + (overstrike) character is not supported. Any carriage control values other than those supported are treated, for the purpose of determining record offset values, the same as the "Space one line" action.

For example, the first 3 records of an input file contain the following. The first character is an ANSI carriage control.

```
1REPORT
-ACCOUNT 777777
0JOHN SMITH
```

Using ASREAD (the default), the indexing parameters to collect the account number and name would be as follows:

```
TRIGGER1=*,2,'REPORT'
FIELD1=1,10,6
FIELD2=2,2,10
```

Using ASPRINTED, the indexing parameters to collect the account number and name would be as follows:

```
TRIGGER1=*,2,'REPORT'
FIELD1=3,10,6
FIELD2=5,2,10
```

Syntax

LINEOFFSET=*value*

Options and values

The value can be:

```
ASREAD
```

ANSI carriage controls are not used to calculate the record offsets for the fields. The offsets are relative to the lines as they are read from the load file.

```
ASPRINTED
```

ANSI carriage controls are used to calculate the record offsets for the fields. The offsets are relative to the line spacing that occurs when the lines are printed.

Related parameters

- “CC” on page 14
- “CCTYPE” on page 14

MCF2REF

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).

Required
No

Default Value
CPCS

Data Type
AFP

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
"RESTYPE" on page 60

MSGDD (Multiplatform)

Determines the name or the full path name of the file where ACIF writes error messages.

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

Required
No

Default Value
stderr

Data Type
AFP, Line

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.

Related concepts
ACIF indexer parameters
Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

MSGDD (z/OS platforms)

Specifies the DD name for the data set to which ACIF writes messages.

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

Required
No

Default Value
SYSPRINT

Data Type
AFP, Line

Syntax

MSGDD=*DD name*

Options and values

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

NEWPAGE

Identifies the skip-to-channel number that indicates a new page in the data stream.

Required
No

Default Value
1

Data Type
AFP, Line

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, and x'F1' - x'F9' and x'C1' - x'C3' in the data if the data were encoded in EBCDIC.

Related parameters

- "CC" on page 14
- "CONVERT" on page 16

OUTCCSID

Specifies a valid coded character set identifier (CCSID) for the output code page you want to have converted.

This parameter can be used by an input record exit program, such as apka2e or asciinpe, to specify the encoding of the output data.

Syntax

OUTCCSID=*ccsid*

Options and values
For *ccsid*, you can specify any valid CCSID, which is a three to five-character decimal value in the range 00000 - 65535 that is registered by the Character Data Representation Architecture (CDRA).

You can replace leading zeros with spaces.

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

OUTEXIT (Multiplatform)

Identifies the name or the full path name of the output record exit program.

Required
No

Default Value
(None)

Data Type
AFP, Line

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, see the discussion in "User exits and attributes of the input file" on page 71.

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.

OUTEXIT (z/OS platforms)

Specifies the name of the output record exit program. This is the module ACIF loads during initialization and subsequently calls for every output record it writes to the output document file (OUTPUTDD).

Required
No

Default Value
(None)

Data Type
AFP, Line

If this parameter is not specified, no output record exit is used.

Syntax

OUTEXIT=*modulename*

Options and values

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

OUTPUTDD (Multiplatform)

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

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

Required
No

Default Value
stdout

Data Type
AFP, Line

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.

Related concepts

ACIF indexer parameters

Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

OUTPUTDD (z/OS platforms)

Specifies the DD name for the output document file ACIF produces when it processes a file.

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

- Variable blocked format
- A maximum record length of 32756

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

This parameter is ignored when you process reports with the ARSLOAD program. If the OUTPUTDD parameter is not specified, ACIF uses OUTPUT as the default DD name.

Required
No

Default Value
OUTPUT

Data Type
AFP, Line

Syntax

OUTPUTDD=*DD name*

Options and values

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

OVLYLIB (Multiplatforms)

Identifies the directories in which overlays are stored.

Required
No

Default Value
(None)

Data Type
AFP

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 UNIX servers, the directory /usr/lpp/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:/opt/IBM/ondemand/V10.5/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 “RESLIB” on page 58.
- USERLIB parameter on page “USERLIB” on page 68.

OVLYLIB (z/OS platforms)

Specifies the data sets that compose the overlay library.

Required
No

Default Value
(None)

Data Type
AFP

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

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

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

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

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

Syntax

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

Options and values

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

Related parameters

- "USERLIB" on page 68

PAGEDEF

Specifies the file name or member name of the page definition.

Required
No

Default Value
(None)

Data Type
AFP

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 line data, mixed-mode data, or unformatted ASCII data into AFP.

The page definition can be located:

- inline within the input file or data set.
- in a user library referenced in the **USERLIB** parameter
- in a library referenced in the **PDEFLIB** parameter.
- in a library referenced in the **RESLIB** parameter (Multiplatform only).

To use an inline page definition:

1. Include an inline page definition in the input file or data set.
2. Specify **CC**=YES to indicate that the input file or data set contains carriage control characters. If the length of the records in the page definition is less than or equal to the logical record length defined for the input file or data set, you can specify fixed length records for the record format:
   - On UNIX and Windows systems:

     ```
     FILEFORMAT=RECORD,n
     ```

     (where n is the logical record length defined for the input file).
   - On z/OS systems, specify record format FBA (fixed block with ANSI carriage control characters) or FBM (fixed block with machine carriage control characters) for the input data set.

   If the length of the records in the page definition is greater than the logical record length defined for the input file or data set, you must specify variable length records:
   - On UNIX and Windows systems:

     ```
     FILEFORMAT=RECORD
     ```

     The first two bytes of each record determine the record length.
   - On z/OS systems, specify record format VBA (variable blocked with ANSI carriage control characters) or VBM (variable blocked with machine carriage control characters) for the input data set.
3. Specify the **PAGEDEF** parameter with one of these values:

- pdefname, which is the name of the inline page definition.

  If the name specified in the **PAGEDEF** parameter does not match the name of an inline page definition, ACIF looks for the page definition in the **PAGEDEF** resource library search path.

  Note: On UNIX servers, the pdefname is case sensitive.
- DUMMY

  If you specify **PAGEDEF**=DUMMY but the file does not include an inline page definition, ACIF looks for the page definition named DUMMY. If ACIF cannot find a page definition named DUMMY, it reports an error and ends processing.

  Note: On z/OS servers, DUMMY must be specified in all uppercase letters.

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 file.

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

On multiplatform systems:

```
PAGEDEF=MEMO
PDEFLIB=/resources
```

On z/OS systems:

```
PAGEDEF=MEMO
USERLIB=USER.RESOURCES
```

Syntax

PAGEDEF=*pdefname*

Options and values

The *pdefname* is the file or member name of the page definition, which is one to eight alphanumeric or national characters, including the two-character prefix if it exists.

Notes

If the file name of the page definition includes a file extension, do not use the file extension when specifying the page definition. However, the file type must be PDEF3820, PDEF38PP, or PDE (or no file type).

For example, to use a page definition named MEMO.PDEF38PP, specify **PAGEDEF**=MEMO.

ACIF does not require a page definition when indexing an AFP data stream file. However, ACIF does require a page definition to transform an input file that contains line data, mixed-mode data, or unformatted ASCII data into MO:DCA-P. If you are transforming such an input file and you do not specify the **PAGEDEF** parameter or you specify **PAGEDEF** without a page definition file name, ACIF reports an error condition and ends processing.

If you use the **PAGEDEF** parameter to specify a page definition that names fonts, but you also use the CHARS parameter to specify fonts, the CHARS parameter is ignored. Therefore, if your page definition names fonts, you should not use the CHARS parameter.

Restriction: z/OS only: ACIF does not support a parameter equivalent to the **LINECT** parameter on the /*JOBPARM, /*OUTPUT, and OUTPUT JCL statements. The maximum number of lines processed on a page is defined in the page definition.

Related parameters

"PDEFLIB" on page 54

PARMDD (Multiplatform)

Identifies the name or the full path name of the file that contains the ACIF parameters, options, and data values.

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

Required
No

Default Value
(None)

Data Type
AFP, Line

Specify the PARMDD parameter only when running ACIF from the command prompt. When you index files using the Content Manager OnDemand data indexing and loading programs, Content Manager 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.

Related concepts
ACIF indexer parameters
Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

PARMDD (z/OS platforms)

Specifies the DD name for the file that contains the ACIF parameters, options, and data values.

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

Required
No

Default Value
SYSIN

Data Type
AFP, Line

Specify the **PARMDD** parameter only when running ACIF from outside of the IBM Content Manager OnDemand load process. When you process files using the IBM Content Manager OnDemand load process, IBM Content Manager OnDemand automatically retrieves the ACIF parameters from the database. If the **PARMDD** parameter is not specified, then ACIF uses SYSIN as the default DD name.

Syntax
PARMDD=*DD name*

Options and values

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

PDEFLIB

On multiplatform systems, **PDEFLIB** identifies the directories in which page definitions are stored. On z/OS systems, **PDEFLIB** specifies the data sets that compose the page definition library.

Required
No

Default Value
(None)

Data Type
AFP

PDEFLIB in a multiplatform environment

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 UNIX servers, the directory /usr/lpp/psf/reslib, if it exists.

PDEFLIB in a z/OS environment

You can specify a maximum of 16 data sets. This parameter also specifies the concatenation sequence when ACIF searches for a particular page definition. ACIF first looks for the resource in dsname1. If ACIF cannot find the resource in dsname1, it continues the search with dsname2, and so on, until it either locates the requested resource or exhausts the list of specified data sets.

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

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

Syntax and options in a multiplatform environment

PDEFLIB=*pathlist*

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

```
PDEFLIB=/tmp:/usr/resources:/opt/IBM/ondemand/V10.5/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.

Syntax and options in a z/OS environment

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

You can specify a maximum of 16 data sets. For example:

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

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

Related parameters

- "PAGEDEF" on page 51
- "RESLIB" on page 58
- "USERLIB" on page 68

PRMODE

If the input data contains shift-in and shift-out codes, determines how ACIF processes them.

Required
No

Default Value
(None)

Data Type
Line, SCS, and Global DJDE

Shift-in and shift-out codes (X'0E' and X'0F') indicate where 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 aaaaaaaa 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
"CHARS" on page 15

PSEGLIB

On multiplatform systems, **PSEGLIB** identifies the directories in which page segments and BCOCA, GOCA, and IOCA objects are stored. On z/OS systems, **PSEGLIB** specifies the data sets that compose the form definition library.

Required
No

Default Value
(None)

Data Type
AFP

PSEGLIB in a multiplatform environment

ACIF searches for page segments in the following order:

1. The paths that you specified with the USERLIB parameter, if any.
2. The paths that you specified with the PSEGLIB parameter, if any.
3. The paths that you specified with the RESLIB parameter, if any.
4. On UNIX servers, the paths that are 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 UNIX servers, the directory `/usr/lpp/psf/reslib`, if it exists.

PSEGLIB in a z/OS environment

Specifies the data sets that compose the page segment library. You can specify a maximum of 16 data sets. This parameter also specifies the concatenation sequence when ACIF searches for a particular page segment or BCOCA, GOCA, or IOCA object. ACIF first looks for the resource in `dsname1`. If it cannot find the resource in `dsname1`, it continues the search with `dsname2`, or further, until it either locates the requested resource or exhausts the list of specified data sets.

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

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

Syntax and options in a multiplatform environment

PSEGLIB=*pathlist*

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

```
PSEGLIB=/tmp:/usr/resources:/opt/IBM/ondemand/V10.5/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.

Syntax and options in a z/OS environment

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

You can specify a maximum of 16 data sets. For example:

```
PSEGLIB=SYS1.PSEGLIB,USER.PSEGLIB
```

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

Related parameters

- "RESLIB" on page 58
- "USERLIB" on page 68

RESEXIT

Identifies the name or the full path name of the resource exit program.

Required
No

Default Value
(None)

Data Type
AFP

This is the program or module ACIF calls each time it attempts to retrieve a requested resource from a directory. If this parameter is not specified, no resource exit is used. For more information about optional program exits you can use to customize how ACIF handles input and output data, see "User exits and attributes of the input file" on page 71.

Syntax

RESEXIT=*name*

Options and values

On multiplatform systems, 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.

On z/OS , *name* is the one- to eight-byte character name of the resource exit program.

RESFILE

Specifies the format of the resource file that ACIF creates.

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

Required
No

Default Value
SEQ

Data Type
AFP

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

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

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

Syntax
RESFILE=*type*

Options and values

The *type* can be:

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

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

Related parameters

- “RESOBJDD (Multiplatform)” on page 59
- “RESOBJDD (z/OS platforms)” on page 59

RESLIB

Determines the paths for the system resource directories.

Required
No

Default Value
(None)

Data Type
AFP

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 UNIX servers, the directory /usr/lpp/psf/reslib, if it exists.
6. On UNIX servers, the directory /usr/lpp/ipfonts, if it exists.
7. On UNIX servers, the directory /usr/lpp/afpfonts, if it exists.
8. On UNIX servers, the directory /usr/lpp/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:/opt/IBM/ondemand/V10.5/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" on page 30
- "FDEFLIB" on page 21
- "OVLYLIB (Multiplatforms)" on page 49
- "PDEFLIB" on page 54
- "PSEGLIB" on page 56
- "USERLIB" on page 68

RESOBJDD (Multiplatform)

Identifies the name or the full path name of the resource file produced by ACIF.

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

Required
No

Default Value
RESOBJ

Data Type
AFP

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.

Related concepts

ACIF indexer parameters
Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

RESOBJDD (z/OS platforms)

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

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

Required
No

Default Value
RESOBJ

Data Type
AFP

Syntax
RESOBJDD=*DD name*

Options and values

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

- Variable blocked format
- A maximum record length of 32756

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

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

RESTYPE

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

Required
No

Default Value
NONE

Data Type
AFP

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** | [**,BCOCA**] [**FDEF**] [**,PSEG**] [**,OVLY**] [**,FONT**] [**,GOCA**] [**,IOCA**] [**,OBJCON**] [**,INLINE**] [**,INLONLY**] [**,PTOCA**] [**,CMRALL**] [**,CMRGEN**]}

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

- **PTOCA**

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

- **CMRALL**

  Specifies that all CMRs required to print or view the output document file (except link CMRs) are included in the resource file. These CMRs include all CMRs referenced in the data stream, all CMRs referenced through a data object or color management resource access table (RAT), and all generic halftone and tone transfer curve CMRs.

- **CMRGEN**

  Specifies that all CMRs referenced in the data stream plus any non-device specific CMRs referenced through a data object or color management RAT (except link CMRs) are included in the resource file. With CMRGEN, the output generated by ACIF is not device specific, unless the data stream explicitly references a device specific CMR.

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

- “CONVERT” on page 16
- “MCF2REF” on page 45
- “RESLIB” on page 58
- “RESOBJDD (Multiplatform)” on page 59
- “RESOBJDD (z/OS platforms)” on page 59

TRACE

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

Required
No

Default Value
NO

Data Type
AFP, Line

Syntax
TRACE=*value*

Options and values

The *value* can be:

NO
ACIF does not produce diagnostic trace records.

YES

On multiplatform systems, ACIF writes trace information to the file specified by the **TRACEDD** parameter.

On z/OS platforms, 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).

TRACEDD (Multiplatform)

Specifies the name or the full path name of the file where ACIF writes trace information when **TRACE**=YES is specified.

Required
No

Default Value
None

Data Type
AFP, Line

Syntax
TRACEDD={TRACE | *filename*}

Options and values

The *filename* is the name or the full path name of the file where ACIF writes trace information when **TRACE**=YES is specified. If you specify the file name without a path, ACIF puts the trace file into your current directory. If **TRACEDD** is not specified, ACIF uses TRACE as the default file name.

TRACEDD (z/OS platforms)

Specifies the DD name of the file where ACIF trace information is written when **TRACE**=PDS is specified.

Required
No

Default Value
None

Data Type
AFP, Line

Syntax
TRACEDD={TRACE | *DD name*}

Options and values

The *DD name* is a 1- to 8-byte character string containing only those alphanumeric characters supported on z/OS systems. The file that is specified must have these characteristics:

```
DCB=(LRECL=121,RECFM=FB,DSORG=PS)
```

If **TRACEDD** is not specified, ACIF uses TRACE as the default *DD name*.

TRC

Identifies whether the input file contains table reference characters (TRCs).

Required
No

Default Value
NO

Data Type
AFP, Line

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

"CHARS" on page 15

TRIGGER

Identifies locations and string values required to uniquely identify the beginning of a group and the locations and string values of fields used to define indexes. You must define at least one trigger and can define up to 16 triggers.

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

Syntax

TRIGGER*n=record,column,value* | **REGEX='regular expression'**[**,(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
If ACIF is using a value, then this is 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 following examples.

Multiplatform systems: If ACIF is using a regular expression, then this is the beginning column where ACIF looks for text that matches the regular expression. The regular expression must match text which begins in the specified column. If a column range is specified, then ACIF will only search the columns in the column range for the text that matches the regular expression. The regular expression must match text which begins in one of the columns specified by the column range. The maximum number of columns to which the regular expression can be applied is 2K (2048 bytes). If there are records in the file which are longer, use a trigger column range to specify a subset of the record.

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 that 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. You can specify either a value or a regular expression, but not both.

REGEX='*regular expression*'
The regular expression that ACIF uses to match the input data. The regular expression must be specified in the code page given by the CPGID parameter, and can be from 1 to 250 bytes in length. The regular expression can be specified in hexadecimal. You can specify either a value or a regular expression, but not both.

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. See the BREAK option of the INDEX parameter for more information about breaking groups.

In Content Manager OnDemand, each indexed group of pages is known as a *document*. When you index an input file and load the data into the system, Content Manager OnDemand stores the group indexes that are generated by ACIF into the database and stores the documents on storage volumes. Content Manager 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.

Related concepts

ACIF indexer parameters
Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

Examples

TRIGGER1

The following TRIGGER1 parameter causes the 400 indexer to search column one of every input record for the occurrence of a start of new page carriage control character. The record value for TRIGGER1 must be an asterisk. 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 the 400 indexer 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 trigger type is optional, but defaults to group. TRIGGER1 must be a group trigger.

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

Group trigger

The following trigger parameter causes the 400 indexer 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 trigger type is optional, but defaults to group.

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

Group trigger with column range
The following trigger parameter causes the 400 indexer 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 trigger type is optional, but defaults to group.

```
    TRIGGER2=6,50:52,X'C1838396A495A340D5A494828599',(TYPE=GROUP)
```

Recordrange trigger
The following trigger parameter causes the 400 indexer to attempt to locate the string value `Account Number` beginning in column fifty within a range of records in each group. The TRIGGER3 string value can occur in records six, seven, or eight following TRIGGER1 in each group.

An asterisk must be used for record number. The 400 indexer uses the recordrange to determine which records to search for the trigger string value. The trigger type is optional, but must be GROUP for a recordrange trigger.

```
TRIGGER3=*,50,X'C1838396A495A340D5A494828599',(TYPE=GROUP,RECORDRANGE=(6,8))
```

Float trigger
The following trigger parameter causes the 400 indexer 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 trigger type is float and must be specified.

```
TRIGGER3=*,5,X'E3A8978540968640C99583969485',(TYPE=FLOAT)
```

Triggers using a regular expression
The following TRIGGER parameter examples use regular expressions to search for strings.

The following trigger parameter causes the 400 indexer to search for the string "PAGE 1". The search will start in column 1 and continue until the end of each record.

```
CPGID=37
TRIGGER1=*,*,REGEX='PAGE 1',(TYPE=GROUP)
```

The following trigger parameter causes the 400 indexer to search for a string containing four uppercase letters followed by three digits. The regular expression must match the text starting in column 10.

```
CPGID=1141
TRIGGER2=*,10,REGEX='[A-Z]{4}[0-9]{3}',(TYPE=FLOAT)
```

The following trigger parameter causes the 400 indexer to search in columns 15 through 18 for a string containing the letter "P" followed by three lowercase letters. The TYPE is GROUP by default.

```
CPGID=500
TRIGGER1=*,15:18,REGEX=X'D74A8160A95AC0F3D0'
/* regular expression is P[a-z]{3} */
```

UNIQUEBNGS

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.

Required
No

Default Value
YES

Data Type
AFP, Line

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
"DCFPAGENAMES" on page 17

USERLIB

On multiplatform systems, **USERLIB** identifies the names of user directories containing AFP resources for processing the input file. On z/OS systems, **USERLIB** specifies data sets containing AFP resources for processing the input data set.

Required
No

Default Value
(None)

Data Type
AFP

USERLIB in a multiplatform environment

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 UNIX servers, the directory /usr/lpp/psf/reslib, if it exists.
6. On UNIX servers, the directory /usr/lpp/ipfonts, if it exists.
7. On UNIX servers, the directory /usr/lpp/afpfonts, if it exists.
8. On UNIX servers, the directory /usr/lpp/psf/fontlib, if it exists.

Syntax and options in a multiplatform environment

USERLIB=*pathlist*

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

```
USERLIB=/tmp:/usr/resources:/opt/IBM/ondemand/V10.5/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.

USERLIB in a z/OS environment

You can specify a maximum of 16 data sets. ACIF dynamically allocates these data sets and searches for resources in them in the order specified in the USERLIB parameter. If a resource is not found, ACIF searches the appropriate resource libraries defined for that resource type (for example, PDEFLIB for page definitions). The libraries you specify can contain any AFP resources (fonts, page segments, overlays, page definitions, or form definitions). If Resource Access Control Facility (RACF) is installed on your system, RACF checks the authority of the user ID requesting access to a user library (data set). If ACIF is not authorized to allocate the data set, it reports an error condition and ends processing.

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

Syntax and options in a z/OS environment

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

You can specify a maximum of 16 data sets. For example:

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

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

USERMASK

Identifies a symbol and string used to match a field.

Required
No

Default Value
(None)

Data Type
AFP, Line

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'
```

Example

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 uppercase or lowercase 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
"FIELD" on page 22

Related concepts
ACIF indexer parameters
Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

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).

Syntax

USERPATH=*pathlist*

Options and values

*Table 4. Additional options and values for the **USERPATH** parameter.* The *pathlist* is any valid search path. You must use a colon (:) on AIX and z/OS systems or a semicolon (;) in Windows to separate multiple paths.

Platform	Value
AIX	`userpath=/jdoe/fonts/truetype:/jdoe/fonts /truetype/myfonts/`
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.

Print 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 Content Manager OnDemand, PSF or Infoprint Manager system programmer. Show your system programmer these messages, because they might not be contained in the Content Manager 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.

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

Related reference

INDXEXIT (Multiplatform)

Identifies the name or the full path name of the index record exit program.

User programming exits

IBM provides several sample programming exits to assist you in customizing ACIF.

On AIX, the sample programs are in `/opt/IBM/ondemand/V10.5/exits/acif`. On Linux, the sample programs are in `/opt/ibm/ondemand/V10.5/exits/acif`. On Windows, the sample programs are in `\Program Files\IBM\OnDemand\V10.5\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**, **INDXEXIT**, **RESEXIT**.

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.

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. 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.

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

The following example 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;
```

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 the attributes of the input file 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 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.

The following example 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
```

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 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.

Remember: 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 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 (Multiplatform)
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 Content Manager 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 as described in the preceding information. 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 Content Manager 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 commands:

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 Content Manager 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 INDXEXIT parameter.

This exit receives control before a record (structured field) is written to the index object file. The exit program can request that the record be ignored or processed. The 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).

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

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

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.

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.

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 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.

The following example contains a sample DSECT that describes the control block that is passed to the exit program.

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

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.

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.

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 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.

The following example 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;
```

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.

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.

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 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.

The following example 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
```

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.

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.

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 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).

The following example 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;
```

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.

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.

The following example 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
```

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.

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 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.

User exit search order

z/OS systems only: When ACIF loads a specified user exit program during initialization, the z/OS operating system determines the search order and method used to locate these load modules.

Exit load modules can reside in a load library that is used as STEPLIB, JOBLIB, or in a system library. ACIF uses the standard z/OS search order to locate the exit load module. ACIF looks first in the STEPLIB, then in the JOBLIB, and finally in the system libraries.

Non-Zero return codes

If ACIF receives a non-zero return code from any exit program, ACIF issues message APK412 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.

The following example shows the format of the data structure in UNIX and Windows in the form of a sample print file attributes C language header.

```
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;
```

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.

1. Each of the previous character values is left-aligned; 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.

The following example shows the format of the data structure in the form of a z/OS sample print file attributes DSECT.

```
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 '
```

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.

1. Each of the previous character values is left-aligned, with padding blanks added following 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.

ACIF exits written in COBOL (z/OS systems)

ACIF is not a COBOL Language Environment aware application so special considerations are needed to use ACIF exits with the COBOL Language Environment.

When you use ACIF to invoke an exit, a Language Environment is not active. The normal behavior for Language Environment is that when the exit is entered, the Language Environment is created, and the Language Environment branches to the actual exit code. When the actual exit code returns, the Language Environment is destroyed and you return to ACIF.

What this means is that every time an exit is invoked, the Language Environment is created and destroyed, causing significant performance overhead. Reinitialization of the environment also reinitializes variables every time the exit is invoked. For COBOL, Language Environment provides a runtime option, RTEREUS(ON), which allows the Language Environment to persist as long as certain programming requirements are met. The environment is not destroyed every time the exit returns, but reused when an exit is reentered. Certain compile options are incompatible with RETREUS(ON). See the *Language Environment Programming Reference* for restrictions and guidelines about the RTEREUS(ON).

Attention: You can specify a RTEREUS(ON) in a number of ways. For example, the CEEOPTS DD. RTEREUS(ON) should only be specified in a manner which only affects the ACIF COBOL exits. If a

CEEOPTS DD is used, it affects all Language Environment programs in that JOB step, like ARSLOAD. Do not use a CEEPRMxx member of PARMLIB.

In order to minimize the scope of RTEREUS(ON), a CEEUOPT CSECT must be assembled and link-edited with the COBOL object code. For more information on constructing a CEEUOPT CSECT, see the *z/OS Language Environment Customization Guide*.

You can use this sample as a model, but you must be sure that the following option is specified:

```
RTEREUS=(ON)
```

Tip: Specify the ALL31(ON) option. You must ensure that the resulting module is link-edited as NOT RE-ENTRANT and NOT REUSABLE. This is required to allow the local variables in the COBOL exit code to retain their values across multiple invocations.

CEEUOPT CSECT sample

```
CEEUOPT CSECT ,
CEEUOPT AMODE ANY
CEEUOPT RMODE ANY

        CEEXOPT
              ABPERC=(NONE),                                          +
              ABTERMENC=(ABEND),                                      +
              AIXBLD=(OFF),                                           +
              ALL31=(ON),                                             +
              ANYHEAP=(16K,8K,ANYWHERE, FREE)                         +
              BELOWHEAP=(8K,4K,FREE),                                 +
              CBLOPTS=(ON),                                           +
              CBLPSHPOP=(ON),                                         +
              CBLQDA=(OFF),                                           +
              CEEDUMP=(60,SYSOUT=*,FREE=END,SPIN=UNALLOC),            +
              CHECK=(ON),                                             +
              COUNTRY=(US),                                           +
              DEBUG=(OFF),                                            +
              DEPTHCONDLMT=(10),                                      +
              DYNDUMP=(*USERID,NODYNAMIC,TDUMP),                      +
              ENVAR=(''),                                             +
              ERRCOUNT=(0),                                           +
              ERRUNIT=(6),                                            +
              FILEHIST=(ON),                                          +
              FILETAG=(NOATOCVT,NOAUTOTAG),                           +
              HEAP=(32K,32K,ANYWHERE,KEEP,8K,4K),                     +
              HEAPCHK=(OFF,1,0,0,0),                                  +
              HEAPPOOLS=(OFF,8,10,32,10,128,10,156,10,1024,10,2048,   +
              10,0,10,0,10,0,10,0,10,0,10,0,10),                      +
              INFOMSGFILTER=(OFF,,,,),                                +
              INQPCOPN=(ON),                                          +
              INTERRUPT=(OFF),                                        +
              LIBSTACK=(4K,4K,FREE),                                  +
              MSGFILE=(SYSOUT,FBA,121,0,NOENQ),                       +
              MSGQ=(15),                                              +
              NATLANG=(ENU),                                          +
              NOAUTOTASK=,                                            +
              NOTEST=(ALL,*,PROMPT,INSPPREF),                         +
              NOUSRHDLR=(''),                                         +
              OCSTATUS=(ON),                                          +
              PC=(OFF),                                               +
              PLITASKCOUNT=(20),                                      +
              POSIX=(OFF),                                            +
              PROFILE=(OFF,''),                                       +
              PRTUNIT=(6),                                            +
              PUNUNIT=(7),                                            +
              RDRUNIT=(5),                                            +
              RECPAD=(OFF),                                           +
              RPTOPTS=(OFF),                                          +
              RPTSTG=(OFF),                                           +
              RTEREUS=(ON),                                           +
              SIMVRD=(OFF),                                           +
              STACK=(128K,128K,ANYWHERE,KEEP,512K,128K),              +
              STORAGE=(NONE,NONE,NONE,OK),                            +
              TERMTHDACT=(TRACE,,96),                                 +
              THREADHEAP=(4K,4K,ANYWHERE,KEEP),                       +
              THREADSTACK=(OFF,4K,4K,ANYWHERE,KEEP,128K,128K),        +
              TRACE=(OFF,4KDUMP,LE=0),                                +
              TRAP=(ON,SPIE),                                         +
```

```
              UPSI=(00000000),                                  +
              VCTRSAVE=(OFF),                                   +
              XPLINK=(OFF),                                     +
              XUFLOW=(AUTO)                                     +

        END         ,
```

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
- 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 example shows 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 example shows 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.

ACIF processing of 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
```

The BNG and TLE Structured Fields cannot be nested. For example, the ARSLOAD program cannot process the following file:

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

```
      EPG
    ENG
  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 Content Manager 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 Content Manager OnDemand.
- If the input file does not contain any BNG - ENG pairs, then the load process may run out of memory looking for the start and end of the groups.

Format of the 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 Content Manager 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. ACIF assigns a null value (X'FFFF') to the Graphic Character Set Global Identifier.

Related concepts

ACIF indexer parameters

Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

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 128 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.

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 following examples illustrate the two possible AFP data stream document formats ACIF may produce.

Example of Code Containing Group-Level Indexing:

```
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
```

The following example illustrates one of the formats (containing group-level and page-level indexing) 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
```

The “Format of the ACIF output document file” on page 93 topic 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 Content Manager 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 field, 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. 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 the format of the ACIF index object file, 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 128 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 data stream

When ACIF produces an output file in the MO:DCA format, each MO:DCA in the file is a single record preceded by a X'5A' carriage control character.

There are 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
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.

Begin Print File (BPF) and End Print File (EPF) Structured Fields
MO:DCA-P data that ACIF processes might contain BPF and EPF structured fields, which define the boundaries of a print file.

The BPF structured field is at the beginning of the MO:DCA-P input file and the EPF structured field is at the end of the file.

Some products concatenate the ACIF resource file (RESOBJDD) to the front of the output file (OUTPUTDD). However, if any data, such as a resource group, is found before the BPF structured field or after the EPF structured field in the ACIF output file, the MO:DCA-P data stream is not valid. By default, ACIF removes the BPF and EPF structured fields from the MO:DCA-P input file before processing the file. Also, if the input file contains an index object, ACIF ignores it and does not pass it to the output file.

The EXTENSION=PASSPF parameter indicates that ACIF must pass the BPF and EPF structured fields to the output file when they are found in the input file. This parameter also verifies whether a BPF/EPF structured field pair that the input record exit tries to insert is actually inserted. If PASSPF is not specified and the input record tries to insert a BPF/EPF pair, the attempt fails and the pair is discarded.

Attention:

1. Be careful using PASSPF. If the output file contains BPF and EPF structured fields and it is concatenated with the resource file, the resulting MO:DCA-P data stream is not valid.
2. When PASSPF is specified, ACIF passes all Begin Document (BDT) and End Document (EDT) structured field pairs from the MO:DCA-P input file to the output data stream without adding the normal comment and timestamp triplets.
3. ACIF issues an error message if PASSPF is specified with the IDXCPGID parameter. If EXTENSIONS=ALL is specified, PASSPF is ignored and IDXCPGID is used.
4. ACIF does not verify whether the input file is MO:DCA IS/3 compliant. Before ACIF discards or passes the BPF and EPF structured fields, it checks the placement and format of the pair in the input file. For example, if the input file contains a BPF structured field, it must also contain an EPF structured field. If the BPF/EPF pair is incorrect, ACIF issues an error message. If the placement and format is correct, ACIF discards or passes the pair.

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.

ACIF examples

Example one: Bank loan 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. The detail records contain several fields, including the loan number. The example also shows a sample page of the loan report, as it appears when viewed using an IBM Content Manager OnDemand client program.

The loan report

```
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        CUSTOMER             LOAN        DELINQUENT    DELINQUENT    DELINQUENT
   NUMBER         NAME              AMOUNT         30 DAYS       60 DAYS       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
```

Each page in the loan report follows the same 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.

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. The example shows the indexer parameters that are required for ACIF to process the loan report.

Report data processing
To process a sample report, you typically create or extract a subset of a complete report.

The example shows how to use the graphical indexer to process a sample report and create indexing information. The graphical indexer is part of the Content Manager OnDemand server and is a program that runs on a Windows workstation. 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, Content Manager OnDemand uses the indexing parameters, options, and data values that are stored with the Content Manager 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, Content Manager OnDemand might not index the data properly. For example, Content Manager OnDemand might not be able to locate triggers, indexes, and fields or extract the correct index values.

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 Content Manager 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
A Content Manager 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.

General page
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 page
The **View Information** page is where you specify information needed by Content Manager 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 page
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. 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 Content Manager OnDemand graphical indexer.

Opening reports
When you load a report into the system, Content Manager OnDemand uses the indexing parameters, options, and data values that are stored with the Content Manager OnDemand application to index the data.

Procedure

To open a report:

1. In the Add an Application window, click the **Indexer Information page**.
2. Select **Sample Data**.
3. Click **Modify**.
4. Select the name of the file that contains the sample data and then click **Open**.

Results
In the **Parameter Source** area, 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 with a warning that the data must match the data that is being loaded.

Restriction: If the data that is being loaded does not match the data that you used to generate indexing parameters with the graphical indexer, Content Manager OnDemand might not index the data properly. For example, Content Manager OnDemand might not be able to locate triggers, indexes, and fields or extract correct index values.

Defining fields

The first field identifies where ACIF locates the date.

About this task

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

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.

Procedure

To define fields:

1. Select the field by clicking the area in the report that contains the field data.

   For example, 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)`. The graphical indexer highlights the value.
2. Right-click the field and select **Field to display the Add a Field**.
3. Save the field information and click **OK** to add the field and return to the report window.
4. Right-click the **Transaction** field to open the **Add a Field** dialog box.

   To support the way that Content Manager OnDemand should segment and load the data and the way users will search for reports, define a transaction field. A transaction field allows Content Manager 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, which results in proper data loading and better retrieval performance.
5. Select the field by clicking the area in the report that contains the field data. Select the loan number that is displayed in column three of the ninth record on the page.

   The loan number is displayed as `0000010000`.

What to do next

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.

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, right-click 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 Content Manager OnDemand should segment and load the data, and the way users will search for reports, define a *transaction field*. A transaction field allows Content Manager 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, right-click 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 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.

Procedure

To define indexes:

1. Clear any selected triggers or fields by clicking a blank area of the report.
2. Click the **Add an Index** icon on the toolbar to open the **Add an Index** dialog box.
3. Accept the suggested default and the application group database field name, `report_date`.

   The Identifier (Index1) determines the name of the index parameter. The Attribute is the name of the index. When data is loaded into the application group, the date index values will be stored in the `report_date` application group database field.
4. Select **Group**. To store data in the system, you must define at least one group index.

   The Type of Index determines the type of index generated by ACIF.
5. Set Break to **Yes**.

   A group index must always control the break. Even though in this example, the group index does not control the break, the GROUPMAXPAGES parameter does.
6. Identify the field parameter that ACIF uses to locate the index and click **Add**.

   The **Fields** area lists the field parameters that were defined (in the Fields list) for the report.
7. Type `Loan Number` in the Attribute field.

   The Identifier is Index2. Later, on the **Load Information** page, you will map the index to application group database fields. Loan number values are not mapped directly to a database field, so you must enter your own index name in the **Attribute** field.
8. Select an index type of **GroupRange** and identify the field parameter that ACIF uses to locate the index.

   ACIF should extract the beginning and ending loan numbers for a group of pages. Because a grouprange trigger can never break a group, Break must always be set to No.
9. Select **Field2** and add it to the Order list. Click **Add** to add the index.
10. Click **Done** to close the **Add an Index** dialog box.

Displaying triggers, fields, and indexes

You can display the triggers, fields and indexes information in the graphical indexer. When you click the icon, the graphical indexer changes to display mode (note the status bar). The triggers and fields appear highlighted.

Procedure

To display triggers, fields, and indexes:

1. 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.
2. Scroll through pages of the report to verify that they appear in the correct location on all pages.
3. Click the **Display and Add Parameters** icon to return to add mode.
4. Click the **Select Trigger, Index, Field Parameters** icon on the toolbar to open the **Select** dialog box.

   By 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.
5. Click **Cancel** and **Trigger1** and then click **Properties** to display the **Update a Trigger** dialog box.
6. Click **Cancel** and then close the**Select** dialog box.

Setting values in the Indexer Properties dialog box

After defining the view information, the triggers, fields, and indexes, complete the indexing information for the loan report by setting values in the **Indexer Properties** dialog box.

About this task

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 can 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.

Procedure

To set values in the **Indexer Properties** dialog box:

1. 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.
2. Click **OK** to save the changes and return to the report window.
3. 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 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 page

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. Content Manager 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: Phone bill

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 phone bill

Return this portion with your payment.

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

Make check payable to

MOUNTAIN COMMUNICATIONS

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

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

TOTAL $56.97

MOUNTAIN COMMUNICATIONS
BOULDER CO 80000-0000

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

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

THANK YOU FOR YOUR PAYMENT

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

SUMMARY OF CURRENT CHARGES

RESIDENCE SERVICE	$25.07
911 SURCHARGE	$0.60
CUSTOMER ACCESS SERVICE	$3.50
WIRING MAINTENANCE PLAN	$0.60
FEDERAL EXCISE TAX	$0.50
STATE TAX	$0.49
LONG DISTANCE CHARGES (ITEMIZED BELOW)	$26.41

LONG DISTANCE CHARGES

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

PAGE 1

Phone Bill Data Stream

```
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
```

Phone bill example ACIF parameters

```
/* 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      */
```

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. “The phone bill” on page 105 shows

an example of a statement viewed with one of the Content Manager OnDemand client programs. The “Phone Bill Data Stream” on page 105 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. The “Phone bill example ACIF parameters” on page 106 shows the ACIF indexer parameters required to process the data on a UNIX operating system.

Accessing the report

The graphical indexer is part of the Content Manager OnDemand administrative client, a program that runs on a Windows workstation.

This example provides instructions about using the Content Manager OnDemand graphical indexer to process a sample report and create indexing information. 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 personal computer 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, Content Manager OnDemand uses the indexing parameters, options, and data values that are stored with the Content Manager 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, Content Manager OnDemand might not index the data properly. For example, Content Manager OnDemand might not be able to locate triggers, indexes, and fields or extract the correct index values.

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 Content Manager 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

A Content Manager 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 page

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 page
The **View Information** page is where you specify information needed by Content Manager 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 Content Manager 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 page
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, 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 Content Manager OnDemand graphical indexer.

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

About this task

Restriction: When you load a report into the system, Content Manager OnDemand uses the indexing parameters, options, and data values that are stored with the Content Manager 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 Content Manager OnDemand might not index the data properly. For example, Content Manager OnDemand might 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.

Procedure

To open the report:

1. From the **Parameter Source** area, select **Sample Data**.
2. Click **Modify**.
3. Select the name of the file that contains the sample data. Click **Open**.

Defining triggers

ACIF uses one or more triggers to determine where to begin to locate index values.

About this task

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 Content Manager OnDemand, the carriage control characters in column one of the data cannot be viewed from the graphical indexer.

About this task

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.

Procedure

To define TRIGGER1:

1. Select any column in the first record.

   When you select a column, the graphical indexer highlights the data.
2. Click the **Trigger** icon on the toolbar to open the **Add a Trigger** dialog box.
3. Click the **Carriage Control** button in the Columns to Search box.
4. Click **OK** to add the trigger and return to the report window.

Locating the account number

The Identifier (Trigger2) determines the name of the trigger parameter. The trigger Type is Group.

About this task

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.

Procedure

To locate the account number:

1. Select the string ACCOUNT NUMBER in the report.
   The graphical indexer highlights the string.
2. Click the **Trigger** icon on the toolbar to open the **Add a Trigger** dialog box.
3. Click **OK** to add the trigger and return to the report window.

Defining fields
ACIF uses fields to determine where to locate index values.

About this task

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
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.

About this task

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.

Procedure

To define the account number field:

1. Select a field by clicking the area in the report that contains the field data.
   For example, select the account number on the first page in the sample report.
2. Click the **Define a Field** icon on the toolbar to open the **Add a Field** dialog box.
3. Click **OK** to add the field and return to the report window.

Defining the customer name field
When you select a field value, include sufficient blank columns (following the name) to generate a field length to hold the longest name that ACIF will encounter in an actual report.

About this task

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.

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.

Procedure

To define the customer name field:

1. Select a field by clicking the area in the report that contains the field data.
   For example, select the customer name on the first page in the sample report.
2. After you select the sample field value, click the **Define a Field** icon on the toolbar to open the **Add a Field** dialog box.
3. Click **OK** to add the field and return to the report window.

Defining the bill date field

The Identifier (Field3) determines the name of the field parameter.

About this task

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.

Procedure

To define the bill date field:

1. Select a field by clicking the area in the report that contains the field data.
   For example, select the billing date on the first page in the sample report.
2. After you select the sample field value, click the **Define a Field** icon on the toolbar to open the **Add a Field** dialog box.
3. 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.

About this task

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

The Identifier (Index1) determines the name of the index parameter.

About this task

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.

Procedure

To define the account number index:

1. Clear any selected triggers or fields by clicking a blank area of the report.
2. Click the **Add an Index** icon on the toolbar to open the **Add an Index** dialog box.
3. Set Break to Yes.

   ACIF should use the account number index to control the group break. The Break area determines whether ACIF closes the current group and begins a new group when the index value changes.
4. Identify the field parameter that ACIF uses to locate the index.

The Fields area lists the field parameters that have been defined for the report.

5. Select **Field1** in the Fields list and then click **Add** to move **Field1** from the **Fields** list to the **Order** list.

   For the account number index values for this example, use Field1.
6. 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.

About this task

The Fields area lists the field parameters that were defined (in the Fields list) for the report. Identify the field parameter that ACIF uses to locate the index.

Procedure

To define the customer name index:

1. Accept the suggested default name of `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.
2. Set Break to No.

   The Break areas determines whether ACIF closes the current group and begins a new group when the index value changes. ACIF does not use the customer name index to control the group break.
3. Select **Field2** in the Fields list and then click **Add** to move **Field2** from the **Fields** list to the **Order** list.
4. Click **Add** to add the index.

Defining the bill date index

The Identifier (Index3) determines the name of the index parameter.

About this task

The Attribute is the name of the index. The Break area determines whether ACIF closes the current group and begins a new group when the index value changes.

Procedure

To define the bill date index:

1. Accept the suggested default name of `bill_date`.

   When a report is loaded into the application group, the billing date index values are 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 extracts a value for each group in the report.
2. Set Break to No.

   ACIF does not use the billing date index to control the group break. The Fields area lists the field parameters that have been defined for the report.
3. Identify the field parameter that ACIF uses to locate the index.

   For the billing data index values, that is Field3.
4. Select **Field3** in the Fields list and then click **Add** to move **Field3** from the **Fields** list to the **Order** list.
5. Click **Add** to add the index.

Displaying triggers, fields, and indexes

Procedure

To display triggers, fields, and indexes:

1. 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. The triggers and fields are highlighted.
2. Scroll through pages of the report to verify that they appear in the correct location on all pages.

   When you finish, click the **Display and Add Parameters** icon to return to add mode.
3. Click the **Select Trigger, Index, Field Parameters** icon on the toolbar to open the **Select** dialog box.

   By 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.
4. Click **Field1** again to highlight the area on the next page.
5. Click **Field2** to highlight the field in the report.
6. Click **Index1** and then click **Properties** to open the **Update an Index** dialog box. Click **Cancel**.
7. Click **Trigger1** and then click **Properties** to open the **Update a Trigger** dialog box.
8. Click **Cancel**. Close the **Select** dialog box.

Setting 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.

About this task

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 can 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.

Procedure

To set indexer properties:

1. Click the **Data Format** tab.

   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.
2. Specify the name of a form definition and page definition and click the **Resource Information** tab.

   The input data will be converted to AFP, and you must specify values on this page. For the sample phone bill report, ACIF should collect form definitions, overlays, and page segments. Check the appropriate boxes in the Resource File Contents area.
3. In the Search Paths area, identify where ACIF can find the resources. Enter the names of the directories where the resources reside.
4. Click **OK** to save the changes and return to the report window.
5. Close the report window.
6. When prompted, click **Yes** to save your changes and return to the **Indexer Information** page.

Defining the application, part 2

View Information page

The **View Information** page is where you specify information needed by Content Manager 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 page
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 to 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 Content Manager 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: Income statement

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 income statement report

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

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

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

ACIF parameters for the 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'E3A89785409686 40C99583969485',(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       */
```

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. “The income statement report” on page 114 shows an income statement viewed with one of the Content Manager 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.

“ACIF parameters for the income statement” on page 114 shows the ACIF indexer parameters required to process the data.

Accessing the sample report

This example provides instructions about using the Content Manager OnDemand graphical indexer to process a sample report and create indexing information.

About this task

The graphical indexer is part of the Content Manager 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, Content Manager OnDemand uses the indexing parameters, options, and data values that are stored with the Content Manager 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, Content Manager OnDemand might not index the data properly. For example, Content Manager OnDemand might not be able to locate triggers, indexes, and fields or extract the correct index values.

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 Content Manager 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 can 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 might or might 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 TYPE=PAGE.

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.

Related reference
INDEX

Identifies the index name, the field or fields on which the index is based, and the type of index ACIF generates.

Defining the application, part 1
AContent Manager 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 page
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 page
The **View Information** page is where you specify information needed by Content Manager 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 page
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 **Indexer 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 Content Manager OnDemand graphical indexer.

Opening the report
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.

About this task

When you load a report into the system, Content Manager OnDemand uses the indexing parameters, options, and data values that are stored with the Content Manager 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, Content Manager OnDemand might not index the data properly. For example, Content Manager OnDemand might 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.

Procedure

To open the report:

1. Parameter Source area, select Sample Data. The client opens the Indexer Properties dialog box.
2. Click **Modify**.
3. Select the name of the file that contains the sample data. Click **Open**.
4. After you click **OK** or **Cancel**, the client loads the input file into the report window.

Defining triggers
ACIF uses one or more triggers to determine where to begin to locate index values

About this task

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. 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
The graphical indexer displays the report as it is viewed in Content Manager OnDemand, you cannot see the carriage control characters in column one of the data.

About this task

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.

Procedure

To define the trigger:

1. Select any column in the first record.
   When you select a column, the graphical indexer highlights the data.
2. Click the **Trigger** icon on the toolbar to open the **Add a Trigger** dialog box.
3. Click the **Carriage Control** icon in the Columns to Search box.
4. Click**OK** to add the trigger and return to the report window.

Locating the statement number

The Identifier (Trigger2) determines the name of the trigger parameter.

About this task

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.

Procedure

To define the trigger:

1. Locate the income statement number and select the string `Income Statement` in the report:
   The graphical indexer highlights the string.
2. Click the **Trigger** icon on the toolbar to open the **Add a Trigger** dialog box.
3. Click **OK** to add the trigger and return to the report window.

Locating the total income

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.

Procedure

To define the trigger:

1. Select the string **Total Income** in the report.
   The graphical indexer highlights the string.
2. Click the **Trigger** icon on the toolbar to open the **Add a Trigger** dialog box.
3. Specify a Record Range.
   The total income value can occur in one of two records, which depend on the number of address lines in a statement.
4. Click **OK** to add the trigger and return to the report window.

Example

The Records to Search area determines the records ACIF searches to locate the trigger. 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.

Locating the type of income

Procedure

To define the trigger used to locate the type of income:

1. Select the string Type of Income in the report.

   The graphical indexer highlights the string. 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.
2. 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.
3. 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.
4. Click **OK** to add the trigger and return to the report window.

Locating the subtotal

Procedure

To define the trigger used to locate the subtotal:

1. Select the string Subtotal in the report.

   The graphical indexer highlights the string. 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.
2. 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.
3. 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. ACIF begins its search in this column (5 in the sample report). The Value area contains the trigger string value that ACIF searches for. The graphical indexer displays the starting column number of the string selected in the report.
4. Click **OK** to add the trigger and return to the report window.

Defining fields

About this task

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

Procedure

To define the statement date field:

1. Select a field by clicking the area in the report that contains the field data.
   For example, select the statement date on the first page in the sample report.
2. 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.
3. Click **OK** to add the field and return to the report window.

Defining the statement number field

Procedure

To define the statement number field:

1. Select a field by clicking the area in the report that contains the field data.
   For example, select the statement number on the first page in the sample report.
2. 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.
3. Click **OK** to add the field and return to the report window.

Defining the total income field

Procedure

To define the total income field:

1. Select a field by clicking 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 (following 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.
2. Select the sample field value and 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.

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

Defining the type of income field

Procedure

To define the type of income field:

1. Select a field by clicking 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 (following 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.
2. Select the sample field value and 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.
3. Click **OK** to add the field and return to the report window.

Defining the subtotal field

Procedure

To define the subtotal field:

1. Select a field by clicking 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 (following 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.
2. Select the sample field value and 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.
3. 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.

About this task

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

Procedure

To define an index, first clear any selected triggers or fields by clicking a blank area of the report:

1. 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. ACIF should extract one date index value for each group in the report, accept the default Type of Group.
2. Set Break to Yes.

   The Break areas determines whether ACIF closes the current group and begins a new group when the index value changes. ACIF should use the statement date index to control the group break. The Fields area lists the field parameters that have been defined for the report.
3. Identify the field parameter that ACIF uses to locate the index.

   For the statement date index values, that is Field1.
4. Select **Field1** in the Fields list and then click **Add** to move **Field1** from the **Fields** list to the **Order** list.
5. Click **Add** to add the index.

Defining the statement number index

About this task

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.

Procedure

To define the statement number index:

1. Accept the default Type of Group.

   ACIF should extract one income statement value for each group in the report.
2. Set Break to Yes.

   The Break area determines whether ACIF closes the current group and begins a new group when the index value changes. ACIF should use the statement number index to control the group break. The Fields area lists the field parameters that have been defined for the report.
3. 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.
4. 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.

Procedure

To define the total income index:

1. Accept the default Type of Group. Identify the field parameter that ACIF uses to locate the index.

   ACIF should extract one total income value for each group in the report.
2. Set Break to No.

   The Break area determines whether ACIF closes the current group and begins a new group when the index value changes. ACIF should not use the total income index to control the group break. The Fields area lists the field parameters that have been defined for the report.
3. 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.
4. 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, Content Manager 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.

About this task

Attribute names should be meaningful to the user. Enter `Type of Income`. The Type determines the type of index ACIF generates. The page indexes are stored with the AFP document. A page index cannot be used to break a group.

Procedure

To define the type of income index:

1. Set the Type to Page.

   ACIF extracts type of income indexes for each page in a statement.
2. Identify the field parameter that ACIF uses to locate the index.

   The Fields area lists the field parameters that were defined for the report.
3. 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.
4. Click **Add** to add the index.

Defining the subtotal index
The Identifier (Index5) determines the name of the index parameter.

About this task

The Attribute is the name of the index. For AFP documents, Content Manager OnDemand displays the attribute names and values of page indexes in the **Go To** dialog box, which allows 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. 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 were defined for the report.

Procedure

To define the subtotal index:

1. Set the Type to Page.

   ACIF extracts subtotal indexes for each page in a statement.
2. Identify the field parameter that ACIF uses to locate the index.
   For the subtotal index values, that is Field5.
3. Select **Field5** in the **Fields** list and then click **Add** to move Field5 from the **Fields** list to the **Order** list.
4. Click **Add** to add the index.
5. 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.

Defining 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.

About this task

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.

Procedure

To define the indexer properties:

1. Click the **Data Format** tab.

   The input data will be stored in the system as AFP data (to support the page-level indexes in the output file).
2. 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.
3. Click the **Resource Information** tab.

   The line data input will be converted to AFP. Specify the name of a form definition and page definition.
4. To collect form definitions, select the appropriate option in the Resource File Contents area.
5. In the Search Paths area, specify the location of the resources. Enter the name of the directories where the resources reside.
6. Click **OK** to return to the report window. Close the report window.
7. When prompted, click **Yes** to save your changes and return to the **Indexer Information** page.

Defining the application, part 2

View Information page

The **View Information** page is where you specify information needed by Content Manager 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 page

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 Content Manager OnDemand to remove them from the data before storing index values in the database. For decimal fields, such as the **Total Income** and **Subtotal** fields, 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: AFP data

This example shows how to create indexing information for an AFP input file that contains Tagged Logical Element (TLE) structured fields.

The AFP document

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

ACIF Parameters for the 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                                                 */
DCFPAGENAMES=YES                            /* unique page names           */
UNIQUEBNGS=YES                              /* unique group names          */
IMAGEOUT=ASIS                               /* leave images alone          */
INDEXOBJ=ALL                                /* required for large object   */

/* RESOURCE INFORMATION                                                 */
FORMDEF=F1A10110                            /* default formdef             */
USERLIB=/opt/IBM/ondemand/V10.5/pubs/reslib /* resource library            */
RESTYPE=FDEF,PSEG,OVLY                            /* collect resources    */
```

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 Content Manager OnDemand client programs. The IBM Document Composition Facility (DCF) is an example of an application that can create AFP files that contain TLE structured fields. “The AFP document” on page 127 shows a page of a sample document viewed with one of the Content Manager OnDemand client programs.

For faster retrieval, the input data should be loaded into the system as Content Manager OnDemand large objects in groups of 20 pages. “ACIF Parameters for the AFP document” on page 128 shows the indexer parameters required for ACIF to process the AFP file on a UNIX operating system.

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 need to be processed with the graphical indexer.

Key concepts
There are several important concepts you need to understand about this example.

Large Object

Provides enhanced usability and better retrieval performance for reports that contain very large logical items, for example, statements that exceed 500 pages, and files that contain many images, graphics, fonts, and bar codes. Content Manager 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 Content Manager 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, which is 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.

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.

TLEs
If an AFP file contains TLEs you cannot specify **TRIGGER**, **FIELD** and**INDEX** parameters because the file already contains index information.

Defining the application, part 1
A Content Manager 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.

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 Content Manager 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 Content Manager 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
Create indexing parameters through the **Add an Application** window.

About this task
The system does not verify the parameters, options, or data values that you type in the **Edit Indexer Parameters** window.

Procedure

To create indexing parameters:

1. From the Parameter Source area, select **Sample Data**.
2. In the Parameter Source area, select **Keyboard**.
3. Click **Modify** to open the **Edit Indexer Parameters** window where you can type, modify, and delete ACIF parameters.

Data format parameters
You must specify specific values to certain parameters when you process input data with ACIF and store AFP data in the system.

The following parameters describe the format of the AFP file. 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 INDEXOBJ 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 Content Manager OnDemand large objects.
- 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: **DCFPAGENAMES**=YES. Also, you must change the value of the **DCFPAGENAMES** parameter.

Defining resource information
Several 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 and the directory that contains resources. If all the resources are contained within the AFP file, a directory is not required, and specify FORMDEF=DUMMY.

- **FORMDEF**=F1A10110
- **USERLIB**=/opt/IBM/ondemand/V10.5/pubs/reslib
- **RESTYPE**=FDEF,PSEG,OVLY

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 can 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
After completing all the updates to the application, click **OK** to add the application.

Procedure

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

Using ACIF in z/OS

You can run ACIF on a z/OS system on which the ACIF programs are installed.

To run ACIF on a z/OS system, see the README file provided with the Content Manager OnDemand product package for more information.

Sample JCL

The example shows a sample JCL to invoke ACIF to process print output from an application.

Sample z/OS JCL to Invoke ACIF

```
//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
```

About the JCL statements

The JCL statements 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.

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.

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
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 following are general syntax rules for parameter files:

- 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

The ACIF application example shows an example of z/OS JCL and ACIF processing parameters used to invoke the ACIF program to index an input file.

Example of a z/OS ACIF Application

```
//job...   JOB ...
//APKSMAIN EXEC PGM=APKACIF,REGION=8M,TIME=(,30)
//*============================================================*
//* RUN APK, CREATING OUTPUT AND A RESOURCE LIBRARY            *
//*============================================================*
//STEPLIB  DD  DSN=APKACIF.LOAD,DISP=SHR
//INPUT DD  DSN=USER.ACIFEX2.DATA,DISP=SHR
//SYSIN DD *
                    /* DATA CHARACTERISTICS                    */
CC = YES                            /* carriage control used   */
CCTYPE = A                          /* carriage control type   */
CHARS = GT15
CPGID = 500                         /* code page identifier    */
                    /* FIELD AND INDEX DEFINITION              */
FIELD1 = 13,66,15                   /* Account Number          */
FIELD2 = 0,50,30                    /* Name                    */
FIELD5 = 4,60,12                    /* Date Due                */
INDEX1 = 'Account Number',field1    /* 1st INDEX attribute     */
INDEX2 = 'Name',field2              /* 2nd INDEX attribute     */
INDEX5 = 'Date Due',field5          /* 5th INDEX attribute     */
                    /* INDEXING INFORMATION                    */
INDEXOBJ = ALL
                    /* RESOURCE INFORMATION                    */
FORMDEF = F1A10110                  /* formdef name            */
PAGEDEF = P1A08682                  /*  pagedef name           */
FDEFLIB = SYS1.FDEFLIB
FONTLIB = SYS1.FONTLIBB,SYS1.FONTLIBB.EXTRA
OVLYLIB = SYS1.OVERLIB
PDEFLIB = SYS1.PDEFLIB
PSEGLIB = SYS1.PSEGLIB
RESFILE = SEQ                       /* resource file type      */
RESTYPE = FDEF,PSEG,OVLY            /* resource type selection */
                    /* FILE INFORMATION                        */
INDEXDD = INDEX                     /* index file ddname       */
INPUTDD = INPUT                     /* input file ddname       */
OUTPUTDD = OUTPUT                   /* output file ddname      */
RESOBJDD = RESLIB                   /* resource file ddname    */
                    /* EXIT AND TRIGGER INFORMATION            */
TRIGGER1 = *,1,'1'                  /* 1st TRIGGER             */
TRIGGER2 = 13,50,'ACCOUNT NUMBER:'  /* 2nd TRIGGER             */
/*
//OUTPUT DD DSN=APKACIF.OUTPUT,DISP=(NEW,CATLG),
//         SPACE=(32760,(150,150),RLSE),UNIT=SYSDA,
//         DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VBM,DSORG=PS)
//INDEX DD DSN=APKACIF.INDEX,DISP=(NEW,CATLG),
//         SPACE=(32760,(15,15),RLSE),UNIT=SYSDA,
//         DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VBM,DSORG=PS)
//RESLIB DD DSN=APKACIF.RESLIB,DISP=(NEW,CATLG),
//         SPACE=(12288,(150,15),RLSE),UNIT=SYSDA,
//         DCB=(LRECL=12284,BLKSIZE=12288,RECFM=VBM,DSORG=PS)
```

```
//SYSPRINT DD DSN=APKACIF.SYSPRINT,DISP=(NEW,CATLG),
//         SPACE=(9044,(5,5),RLSE),UNIT=SYSDA,
//         DCB=(BLKSIZE=9044,RECFM=VBA,DSORG=PS)
```

z/OS libraries
The example ACIF parameters define the following libraries.

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 shows the ACIF job created the output files.

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: • ACIF parameters used • Resources used • Return code	APKACIF.SYSPRINT

Concatenating files
Before the Content Manager OnDemand data loading programs can process the z/OS 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 Content Manager OnDemand.

You can then transfer the concatenated file to a Content Manager OnDemand server and process it with the Content Manager OnDemand data loading programs.

The example shows that you can use JCL 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)
```

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.

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 Content Manager 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
- Regular expressions

Control statements that contain numbered lines

You sometimes can receive unexpected results when data set names are continued and the control statements have line numbers in columns 73 - 80 because ACIF reads all 80 columns of the control statements for processing purposes.

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

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.

File transfer

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.

ACIF needs to know the following information to print it:

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

Structured fields are similar to print commands. 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 charactersCode points from X'00' to X'7F', to determine if a file is encoded in ASCII or EBCDIC. If no non-ASCII characters are found, ACIF assumes the file uses the ASCII newline character, X'0A'. Otherwise, ACIF assumes the file uses the EBCDIC newline character, X'25'. Because an input file can misguide ACIF, either intentionally or by accident, a set of rules has been established to determine how ACIF will interpret how a file will be processed. The following table lists the possible combinations.

    Table 7. 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'

    Important: 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 that use 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.

ANSI and machine carriage controls

In many environments (including IBM mainframes and most minicomputers), printable data normally contains a carriage control character.

The carriage control character acts as a vertical tab command to position the paper at the start of a new page, at a specified line on the page, or to control skipping to the next line. The characters can be one of two types: ANSI carriage control or machine carriage control.

ANSI carriage control characters

- The most universal carriage control is ANSI, which consists of a single character that is a prefix for the print line. The following table lists the standard ANSI controls.

Table 8. Standard ANSI controls

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 beginning 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. The following table lists the typical machine controls.

Table 9. 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 (beginning 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 Content Manager 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 Content Manager OnDemand servers and produce output that ACIF can use are:

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

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.

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 binary file.

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.)

Restriction: 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,AFP, or LIST3820 data streams) as a special case. You can always transfer such a file as a binary file 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.

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 Content Manager 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:

- For z/OS , the default is a binary number.
- For files with fixed-length records, binary numbers are recommended (you must know the record length).
- For files with variable-length records that contain only printable characters and either ANSI carriage control characters, or no carriage control characters:
  - Use ASCII and CRLF
  - Specify the control statement `INPEXIT=asciinpe` to remove the otherwise unprintable carriage return (X'0D') that is inserted in the file.
- For VSE files, additional file transfer parameters are available.
- For files with machine carriage control, you can specify BINARY, CRLF and CC. This provides an EBCDIC file with correct carriage controls separated by ASCII newlines and carriage returns.

FTP
From most systems, FTP works similarly to PC file transfer, and most of the same options are provided.

Also, when executing FTP on a Content Manager 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 Content Manager 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 a Content Manager 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.

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 structured field was invoked. The retrieval system can use this information to dynamically create IMM structured field 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:

- 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 Content Manager 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 parmdd=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.

Using regular expressions

A regular expression is a pattern that is used to match characters in a string. There are many online resources that explain the syntax rules of regular expressions.

Regular expression examples

Restriction: Regular expressions are not available on z/OS systems.

The following examples show some common regular expressions:

Table 10. Common regular expressions

Regular expression	Results
Account	Finds the characters "Account." By default searches are case sensitive.
[A-Z]	Finds one uppercase letter.
[A-Z]{3}	Finds three consecutive uppercase letters.

Table 10. Common regular expressions (continued)

Regular expression	Results
[0-9]{5}	Finds five consecutive digits.
[0-9]+	Finds one or more digits.
[^a-z]	Finds everything except lowercase a to z.
\s	Finds one whitespace character (space, tab, and so on).
\S	Finds any character except for whitespace.

ACIF can use a regular expression in the TRIGGER and FIELD parameter. In the TRIGGER, the regular expression specifies the pattern for which to search; in the FIELD, the regular expression is applied to the characters which have been extracted from the field in a way similar to using a mask.

The regular expression must be specified in the code page given by the CPGID parameter. If you are running on an ASCII platform and the CPGID of the document is ASCII then the regular expression can be specified as text, for example:

```
CPGID=819
TRIGGER1=*,*,'PAGE',(TYPE=GROUP)
TRIGGER2=*,25,REGEX='[A-Z]{3}-[A-Z]{6}',(TYPE=FLOAT)
FIELD1=0,9,2,(TRIGGER=1,BASE=TRIGGER)
FIELD2=0,38,10,(TRIGGER=2,BASE=0,REGEX='[A-Z] [0-9]{3}-\S+')
INDEX1='Page',FIELD1,(TYPE=GROUP,BREAK=YES)
INDEX2='Sub-Source',FIELD2
```

In this example TRIGGER2 uses a regular expression, which specifies a pattern of three uppercase letters, followed by a hyphen, followed by six uppercase letters. The text "SUB-SOURCE" would match the pattern.

FIELD2 uses a regular expression, which specifies one uppercase letter, followed by a space, followed by three numbers, followed by a hyphen, followed by one or more non white space characters. The characters "Q 010-1", "I 000-RS", or "L 133-1B" would match this regular expression.

If you are running on an ASCII platform and the CPGID parameter of the document is not ASCII then the regular expression must be specified in hexadecimal in the code page given by the CPGID parameter, for example:

```
CPGID=500
TRIGGER1=*,1,REGEX=X'4AF060F95AC0F3D0'  /* [0-9]{3} */
```

Performance

All text to which the regular expression is applied is converted to UTF-16.

- Performance might not be as fast when you use a regular expression. Using a text string can be faster.
- If the CPGID value is incorrect, the conversion might fail with error message APK2080.

If the regular expression is invalid, ACIF will fail with error message APK484.

Regular expressions and the TRIGGER parameter

On the TRIGGER parameter use the regular expression instead of a text string. A regular expression can be used on both a group trigger and a floating trigger.

The maximum length of the regular expression is 250 bytes.

If an asterisk is specified for the column, ACIF searches the entire record for the string that matches the regular expression. If a column is specified, ACIF searches the text starting in that column for the string that matches the regular expression. The regular expression must match text which begins in that column. If a column range is specified, ACIF searches only the text within the column range for the string

that matches the regular expression. The regular expression must match text which begins in one of the columns specified by the column range.

The maximum record length to which the regular expression can be applied is 2K (2048 bytes). If there are records in the file which are longer, use a trigger column range to specify a subset of the record.

When the regular expression matches the text in a record, ACIF looks for the next trigger, or, if all the group triggers have been found, ACIF collects the fields.

Regular expressions and the FIELD parameter

On the FIELD parameter use the regular expression instead of a mask. A mask and a regular expression cannot both be specified on the same FIELD parameter.

The maximum length of the regular expression is 250 bytes.

The regular expression can be specified on a field based on a group trigger, a field based on a floating trigger, or a transaction field. Masks can be specified only on fields based on floating triggers and transaction fields.

ACIF extracts the text specified by the column and length values. The maximum length of a field that can be specified in the FIELD parameter is 250 bytes. After the field is extracted, ACIF applies the regular expression to the text. Any text that matches the regular expression is extracted for the field. If the matching text is shorter than the length specified in the FIELD parameter, it is padded with blanks until it equals the length.

Default values for fields

If the regular expression does not match any text in the field, a default value can be used. Whether a default value is used and which type of default value depends on the type of field. There are three types of fields: fields based on group triggers, fields based on floating triggers, and transaction fields.

Group field

1. If a regular expression does not match any text in the group field, the default value specified on the FIELD parameter is used. If no default value is specified, ACIF ends with error message APK488.
2. If the record is only long enough to contain part of the field, the regular expression is applied only to the portion of the record that is present.
3. If the record is not long enough to contain even the first byte of the field, the default value specified on the FIELD parameter is used. If no default value is specified, ACIF ends with error message APK449.

Floating field

1. If a regular expression does not match any text in the floating field, there is no error and the default value specified on the FIELD parameter is not used.
2. If the record is only long enough to contain part of the field, the regular expression is applied only to the portion of the record that is present.
3. If the record is not long enough to contain even the first byte of the field, the default value specified on the FIELD parameter is used. If no default value is specified, ACIF ends with error message APK449.
4. In the case of (1) the load process can use the default value in the Application. The other case where the load process uses the default value in the Application is when a floating trigger is not found within a group. Since the trigger is not found, there is no field for that group.

Transaction field (grouprange or pagerange field)

1. If the regular expression does not match any text in the transaction field, there is no error and processing continues. A default value cannot be specified for a transaction field.
2. If the record is not long enough to contain the entire field, no field is collected. There are no errors and processing continues.

Examples

Using a regular expression for a trigger:

```
TRIGGER1=*,1,REGEX='P[A-Z]{3} ',(TYPE=GROUP)
```

This regular expression will match text that begins in column 1 with the letter 'P' and is followed by three uppercase letters followed by a space. For example:

```
"PAGE "
```

Using a regular expression to extract a date in the form of "July 4, 1956":

```
TRIGGER1=*,1,'1'
FIELD1=0,13,18,(REGEX='[A-Z][a-z]+ [0-9]+, [0-9]{4}',DEFAULT='January 1, 1970')
INDEX1='Date',FIELD1
```

Using a regular expression with a transaction field to extract a range of Social Security numbers:

```
TRIGGER1=*,1,'1'
FIELD1=0,30,3
FIELD2=*,*,12,(OFFSET=(59:70),ORDER=BYROW,REGEX='[0-9]{3}-[0-9]{2}-[0-9]{4}')
INDEX1='DEPT',FIELD1,(TYPE=GROUP)
INDEX2='SOCIAL SECURITY NUMBER',FIELD2,(TYPE=GROUPRANGE)
```

Chapter 3. 390 indexer

You can use the 390 indexer to extract index data from and generate index data about line data and AFP reports. In addition, other data types, such as TIFF images, can be captured using the ANYSTORE Exit.

The 390 indexer extracts indexes and stores documents in a single pass of reading the input data. The 390 indexer indexes reports based on the organization of the data in the report. The 390 indexer processes two input sources:

- Indexing parameters that specify how the data should be indexed. You can create the indexing parameters when you define a Content Manager OnDemand application. The parameters are of the same form as used by ACIF, along with some extensions which are unique to the 390 indexer.
- The print data stream.

The 390 indexer indexes input data based on the organization of the data:

- AFP reports. For AFP reports, the index values are already specified within the AFP data stream.
- Document organization. For reports made up of logical items, such as statements, policies, and invoices. The 390 indexer can generate index data for each logical item in the report.
- Report organization. For reports that contain line data with sorted values on each page, such as a transaction log or general ledger. The 390 indexer can divide the report into groups of pages and generate index data for each group of pages.
- Anystore Exit. This exit determines the content and index values of each document.
- Large Object. Large object support is designed to provide enhanced usability and better retrieval performance for reports that contain very large documents by segmenting the documents into groups of pages and downloading only the page groups that the users request to view.

Before you can index a report with the 390 indexer, you must create a set of *indexing parameters*. The indexing parameters describe the physical characteristics of the input data, identify where in the data stream that the 390 indexer can locate index data, and provide other directives to the 390 indexer. Collecting the information needed to develop the indexing parameters requires a few steps. For example:

1. Examine the input data to determine how users use the report, including what information they need to retrieve a report from the system (indexing requirements).
2. Create parameters for indexing.

You run the 390 indexer as part of the Content Manager OnDemand load process with the ARSLOAD program. The Content Manager OnDemand application retrieves the indexing parameters from the Content Manager OnDemand database and uses the parameters to process the input data.

The 390 indexer can logically divide reports into individual items, such as statements, policies, and bills. You can define up to 128 index fields for each item in a report.

The 390 indexer supports input that can be greater than 2 GB, however an individual document inside the input stream can not be larger than 2GB. The maximum size of the input file and documents in a particular environment may be smaller than 2 GB because of the available hardware and any other limitations of the operating environment:

1. If the document (or large object segment) size exceeds 20 MB, then the document data is temporarily stored in the Content Manager OnDemand temporary HFS directory (described in the following information). Therefore, if the largest document is 2 GB, then the temporary HFS directory must have at least 2 GB of available space. If the available HFS disk space is not sufficient to store the largest document in the report, the load fails.

   If the available HFS disk space is not sufficient to store the largest document in the report, the load fails.

   The temporary HFS directory is defined by one of these options:

- The `-c` option in the ARSLOAD parameters. If this is not specified, then:
- The environment variable ARS_TMP. If this is not specified, then:
- The environment variable TEMP. If this is not specified, then:
- The current working directory.

2. In the final load stage, the complete document (or large object segment) needs to be loaded into memory. Therefore, if the document (or large object segment) is 2 GB in size, then the load program needs to be able to acquire 2 GB of memory to load the data. If the available memory is not sufficient to store the largest document in the report, the load fails.

Any data type can be captured using the 390 indexer. Native support exists for line data and AFP data. Other data types, such as PDF and TIFF images, can be captured by using the Anystore Exit. This provides a method to capture documents of any type and size (including those greater than 2 GB) into Content Manager OnDemand.

Indexing

Indexing parameters include information that allow the 390 indexer to identify key items in the input data stream so they can be extracted from the report and stored in the Content Manager OnDemand database. Content Manager OnDemand uses these index values for efficient, structured search and retrieval.

The 390 indexer uses the following methods to determine the index values for each document within a report.

- AFP Reports. The 390 indexer can capture fully resolved AFP data streams (AFPDS). The AFPDS must contain the index values either in the form of TLE or NOP records. For details on these record types, see "INDEXSTYLE" on page 158.

  You can capture AFP resources in either of the following ways:

  – The resources are in-stream at the beginning of the AFPDS. In this case, the Begin Resource Group (BRG) record and End Resource Group (ERG) record must occur prior to the Begin Document (BDT) record.
  – In a z/OS environment, the resources are in a separate input file and specified in the ARSLOAD JCL via a RESOURCE ddname. On AIX, the resources must be included inline at the beginning of the load file.

  In either case, only resource records beginning with the BRG record and ending with the ERG record are captured and stored in the Content Manager OnDemand database.
- Line Print Reports. Line Print Reports consist of text formatted print streams. Column one of each record contains a carriage control character.

  You specify the index information that allows the 390 indexer to segment the print stream into individual items called *groups*. A group is a collection of one or more pages. You define the bounds of the collection, for example, a bank statement, insurance policy, phone bill, or other logical segment of a report file. A group can also represent a specific number of pages in a report. For example, you might decide to segment a 10,000 page report into groups of 100 pages. The 390 indexer creates indexes for each group. Groups are determined when the value of an index changes (for example, account number) or when the maximum number of pages for a group is reached.

  An indexing parameter is made up of an *attribute name* (for example, Customer Name) and an *attribute value* (for example, Earl Hawkins). The parameters include pointers that tell the 390 indexer where to locate the attribute information in the data stream. For example, the tag `Account Number` with the pointer `1,21,16` means that the 390 indexer can expect to find Account Number values starting in column 21 of specific input records. The 390 indexer collects 16 bytes of information starting at column 21 and adds it to a list of attribute values found in the input. For each group that is identified by the 390 indexer, a set of index values that are associated with the group are stored by the Content Manager OnDemand load process into the Content Manager OnDemand database.
- Anystore Exits. The use of an Anystore Exit allows for the capture of any type of data. The exit is responsible for reading the data to be captured, breaking it into documents, and determining the index

values. A sample Anystore Exit is provided which captures TIFF images using a pre-generated set of indexing instructions read from a separate file.

- Large Object. Provides enhanced usability and better retrieval performance for reports that contain very large logical items (for example, statements that exceed 500 pages) and files that contain many images, graphics, fonts, and bar codes. Content Manager OnDemand segments data into groups of pages, compressed inside a large object. You determine the number of pages in a group. When the user retrieves an item, Content Manager OnDemand retrieves and uncompresses the first group of pages. As the user navigates pages of the item, Content Manager OnDemand automatically retrieves and uncompresses the appropriate groups of pages. To enable large object support, fill the **Large Object** check box on the Load Information tab of the Application definition.

  The Large Object option is supported for AFP reports as well as Line Print reports.
- The 390 indexer also provides support for line print reports with global and/or local Xerox DJDE records. These documents can be loaded in the same manner as the standard line print reports described earlier with the addition of DJDE record handling logic. The global DJDE records are stored separately from the individual documents and retrieved at print time as required.

390 indexer parameters

To provide the segmentation and indexing instructions, the 390 indexer process uses a set of indexing parameters. Some parameters are identical to their ACIF indexer counterparts. A subset of those parameters have unique sub-parameters or behave differently than they do in ACIF. If a 390 indexer parameter is similar to an ACIF counterpart, they can be generated using the graphical indexer from the administration client. There are also parameters that are unique to the 390 indexer.

AFPINDEXBUF

Reports that use **INDEXSTYLE=AFP** can have multiple index rows generated per document. This is accomplished either by using multiple TLE records with the same attribute name, or by using the NOP indexing technique with ODZOSDIR records.

The 390 Indexer uses a memory buffer to track the multiple index rows for a given segment. The **AFPINDEXBUF** allows for controlling the size of this buffer. Specify a value representing the number of index rows to put in each buffer.

If the specified value is smaller than the number of index rows for a given document, additional buffers are allocated until enough are created to hold all the index rows for the document. If a report contains documents with a large number of index rows, performance can be improved by setting **AFPINDEXBUF** to a large value. Doing this minimizes the time spent allocating memory. If a report contains documents with only a small number of index rows, you can minimize memory use by keeping the value of **AFPINDEXBUF** small.

Parameters

The default value is 10. The range of valid values is 1 to 10 000. Do not include any punctuation after the equal sign, for example, AFPINDEXBUF=8000.

ANYEXIT

The Anystore Batch Capture Exit can be used to provide all segment and index data to the report capture program. The exit is called dynamically during the capture process. The capture program calls the exit when the indexing instructions for the application include the ANYEXIT parameter. The report administrator provides a program name for the Anystore Exit.

The report capture program expects the Anystore exit to pass back all segment data and the associated index information. The capture program will perform only the data management functions required for the capture process (document compression, document store, index management and store, etc.)

A sample COBOL exit is provided in ARSEXANY along with the COBOL copybook ARSANYBK. A sample C exit is provided in ARSECANY along with the C header files ARSANYBH and ARSZ390H.

Parameters

The following example shows the parameters that are required by **ARSEXANY**.

The parameters can be found in the Cobol Copybook member **ARSANYBK**.

```
01  ANY-HEADER-RECORD.
    02 ANY-APPLICATION-NAME                     PIC X(60).
    02 FILLER.
       05 ANY-FIELD OCCURS 32 TIMES             PIC X(256).
    02 FILLER.
       05 ANY-DISPL OCCURS 32 TIMES             PIC S9(4) COMP.
    02 FILLER.
       05 ANY-LEN  OCCURS 32 TIMES              PIC S9(4) COMP.
    02 ANY-DDNAME                               PIC X(8).
01  ANY-DOC-SPACE.
    02  CURRENT-DOC-BUFFER OCCURS 104832        PIC X.
01  ANY-DOC-SIZE                                COMP PIC S9(8).

01  ANY-STATUS                                  PIC XXX.
    88  ANY-EOF                                 VALUE 'EOF'.
    88  ANY-OVERFLOW                            VALUE 'OVF'.
    88  ANY-INDEX                               VALUE 'IDX'.
    88  ANY-SEG                                 VALUE 'END'.
    88  ANY-ERR                                 VALUE 'ERR'.

01  ANY-IDX-FOUND.
    05 ANY-IDX-FOUND-SET                        PIC X.
       88 ANY-IDX-FOUND-SET-YES                 VALUE 'Y'.
       88 ANY-IDX-FOUND-SET-NO                  VALUE 'N'.
    05 ANY-IDX-FOUND-TABLE.
       10 ANY-F-FIELD OCCURS 32 TIMES.
          15 ANY-F-FIELD-FOUND                  PIC X.
             88 ANY-F-FIELD-FOUND-YES           VALUE 'Y'.
             88 ANY-F-FIELD-FOUND-NO            VALUE 'N'.
01  ANY-INDEX-NAME-COUNT-ORIGINAL               PIC S9(8) COMP.

01  ANY-INDEX-NAMES.
    05 ANY-INDEX-NAME-COUNT-UPDATED             PIC S9(8) COMP.
    05 ANY-INDEX-NAME-TABLE OCCURS 32 TIMES.
       10 ANY-INDEX-NAME                        PIC X(250).

01 ANY-FILE-NAME PIC X(1023).

01 ANY-OS-LEVEL-FLAG PIC X.
   88 ANY-OS-LEVEL-ZOS VALUE 'Z'.
   88 ANY-OS-LEVEL-MP VALUE 'M'.
01 ANY-EBCDIC-ASCII-SWITCH PIC S9(8) COMP.
   88 ANY-EBCDIC-CP VALUE 1.
   88 ANY-ASCII-CP VALUE 0.

01 ANY-INDEXES-33.
   05 ANY-KEY33-IND PIC X(5).
   05 ANY-FIELDS33.
      10 ANY-FIELD33 OCCURS 96 TIMES PIC X(256).
      10 ANY-DISPL33 OCCURS 96 TIMES PIC S9(4) COMP.
      10 ANY-LEN33 OCCURS 96 TIMES PIC S9(4) COMP.
   05 ANY-IDX-FOUND-TABLE33.
      10 ANY-F-FIELD33 OCCURS 96 TIMES.
         15 ANY-F-FIELD-FOUND33 PIC X.
            88 ANY-F-FIELD-FOUND33-YES VALUE 'Y'.
            88 ANY-F-FIELD-FOUND33-NO VALUE 'N'.
   05 ANY-INDEX-NAME-TABLE33 OCCURS 96 TIMES.
      10 ANY-INDEX-NAME33 PIC X(250).
```

ANY-APPLICATION-NAME

Set by capture program. The value of the Application Name.

ANY-DDNAME

Set by capture program. The DD Name of the input data file. This is particularly useful when the load process is initiated by the Spool Capture facility, because the DD Name changes each time.

ANY-DISPL

Set by capture program. The column displacement value. This field represents an array of values that correspond with the **Field** parameter value.

ANY-DOC-SIZE

Set by exit. The size of the current segment that is to be passed to the capture program.

ANY-DOC-SPACE
Set by exit. The segment buffer to be passed to the capture program. Maximum size of 1048320 allowed. If larger sizes are required, the exit must break them into multiple physical segments no larger than 1048320.

ANY-EBCDIC-ASCII-SWITCH
Set by calling routine. This flag indicates whether the code page specified in the indexing parameters (or defaulted to if none was specified) is an EBCDIC or ASCII code page.

ANY-F-FIELD-FOUND
Set by exit. ANY-F-FIELD-FOUND-YES is set on by the exit for each index value returned.

ANY-FIELD
Set by exit. The values for the 32 Keys.

ANY-FILE-NAME
Set by calling routine. The path name of the input file. This is provided instead of the ANY-DDNAME when the load file is coming from a hfs or zfs path.

ANY-INDEX-NAME
Set by both capture program and exit. Contains the name field from the INDEX indexing parameters. The exit can interrogate this array to determine which position in the ANY-FIELD array represents which index value. If the exit adds additional index field names not present in the INDEX indexing parameters, this array must be updated to give the field name. This must match the Application Group field name.

ANY-INDEX-NAME-COUNT-ORIGINAL
Set by capture program. The number of indexes identified by the **INDEX** indexing parameters.

ANY-INDEX-NAME-COUNT-UPDATED
Set by both capture program and exit. If the exit adds additional index field names not present in the **INDEX** indexing parameters, this count must be updated by the exit to indicate the new total number of indexes.

ANY-INDEXES-33
Set by both capture program and exit. This structure is for index fields 33 - 128. If used, the *ANY-KEY33-IND* variable must be set to hexadecimal values of D2C5E8F3F3 (KEY33 in EBCDIC). If any other value is found in *ANY-KEY33-IND*, then this structure is ignored. It is initialized to blanks the first time the exit is called, even if **ANY-INDEX-NAME-COUNT-ORIGINAL** is greater than 32.

ANY-IDX-FOUND-SET
Set by capture program. **ANY-IDX_FOUND-SET-NO** is set on by the calling program before calling this exit.

ANY-LEN
Set by capture program. The field length value. This field represents an array of values that correspond with the **Field** parameter value.

ANY-OS-LEVEL-FLAG
Set by calling routine. This flag indicates whether the load is being performed in a z/OS or multiplatform environment.

ANY-STATUS
Set by exit. Processing status to be passed back to the capture program.

Possible values are:

EOF
End of run; no data nor indexes returned.

OVF
Data returned with more to follow; no indexes returned.

IDX
Indexes returned; no data.

END
Data and indexes returned; end of this document.

ERR
An unexpected error occurred; end the run.

Sample scenarios

1. Document is .5 MB in size, one set of indexes:

   ANY-STATUS = END
   Both the data and the index values are returned.
2. Document is .5 MB in size, three sets of indexes for this document:

   ANY-STATUS = END
   Data and first set of indexes returned.

   ANY-STATUS = IDX
   No data, second set of indexes returned.

   ANY-STATUS = IDX
   No data, third set of indexes returned.
3. Document is 2.5 MB in size, one set of indexes:

   ANY-STATUS = OVF
   First MB of data returned, no indexes returned.

   ANY-STATUS = OVF
   Second MB of data returned, no indexes returned.

   ANY-STATUS = END
   Remainder of data and indexes returned.
4. Document is 2.5 MB in size, three sets of indexes for this document:

   ANY-STATUS = OVF
   First MB of data returned, no indexes returned.

   ANY-STATUS = OVF
   Second MB of data returned, no indexes returned.

   ANY-STATUS = END
   Remainder of data returned, first set of indexes returned.

   ANY-STATUS = IDX
   No data returned, second set of indexes returned.

   ANY-STATUS = IDX
   No data returned, third set of indexes returned.

Eventually, an **ANY-STATUS** of EOF is returned. At that time, no data or any indexes are returned.

Notes

- ANY-IDX-FOUND-SET-NO is set on by the calling program.
- After the ANYSTORE exit returns control, if **ANY-IDX-FOUND-SET-NO** is still on, then the calling routine will act as though the **ANY-F-FIELD-FOUND-YES** and **ANY-F-FIELD-FOUND33-YES** flags are on for each valid index field. This behavior allows the process to work as it did before this change was added to version 8.5, so existing ANYSTORE exits do not need to be modified.
- If the exit turns on the **ANY-IDX-FOUND-SET-YES** flag, then the **ANY-F-FIELD-FOUND-YES** and **ANY-F-FIELD-FOUND33-YES** flags must be set on for each index being returned.
- **ANY-INDEX-NAME-COUNT-ORIGINAL** and **ANY-INDEX-NAME-COUNT-UPDATED** are both set before each time the ANYSTORE exit is called. They are set to the number of **INDEX** indexing parameters present in the application definition.
- If the exit is adding an index value to the **ANY-FIELD** for **ANY-FIELD33** arrays for an application group field for which there is no **INDEX** indexing parameter, then the **ANY-INDEX-NAME-COUNT-UPDATED** field must be set to the new total number of indexes. The name of the index field, as defined in the application group, must be added to the **ANY-INDEX-NAME** or **ANY-INDEX-NAME33** array.

Developing an ANYSTORE exit
The sample exit routine stores TIFF data and also creates a second index (for illustration purposes) for each logical document.

Procedure

The following approach is recommended for developing an ANYSTORE exit:

1. Create report definition with an ANYSTORE exit name specified in the ANYEXIT parameter.
2. Create a new member in the Sample library corresponding to the ANYSTORE exit name.
3. Copy the sample exit into the new member.
4. Modify exit as required.
5. Compile and link the exit.
6. Test.

BREAKYES

A break is a condition that exists in the load file to determine whether the current page is the start of a new document.

The following indexing parameters determine breaks:

- TRIGGER parameters that use **TYPE=GROUP**. These parameters are used as a match condition. The trigger value must be found on this page for a break condition to exist.
- INDEX parameters that use **BREAK=YES** These parameters are used as a change condition. The index value must change on this page from the value found on the previous page for a break condition to exist. However, if the index value contains multiple fields which are to be concatenated together, only the first field is examined to determine if a break condition exists.

When multiple break conditions exist, regardless of whether they are **TRIGGER** parameters with **TYPE=GROUP** or **INDEX** parameters with **BREAK=YES**, they are all logically connected by using AND or OR operators based on the **BREAKYES** parameter.

Parameters

When BREAKYES=AND is specified
ALL triggers with TYPE=GROUP must be found and ALL index values with BREAK=YES must change for the current page to be considered the start of a new document.

When BREAKYES=OR is specified
At least one TRIGGER with TYPE=GROUP must be found or one INDEX with BREAK=YES must change for the current page to be considered the start of a new document.

The default value is BREAKYES=AND
If a value other than AND or OR is specified for this parameter, a warning message is issued, and the parameter is interpreted as though BREAKYES=AND was specified.

CPGID

Specifies the code page of the index values being extracted from the documents. Default value is 500.

The index values are translated from this code page into the code page of the database at report capture time. The code page identifier is used by the Content Manager OnDemand client programs to display indexing information.

DJDECNT

This integer field identifies the number of global DJDE records that are to be expected before the first document is found. This parameter is used in conjunction with the DJDETRIG and DJDECOL parameters.

`DJDECNT=n`

Where n is greater than or equal to 0. For example, `DJDECNT=4`.

DJDECOL

This integer field identifies the column (including the carriage control column) in which the DJDE identifier can be found. This parameter is used in conjunction with the DJDETRIG and DJDECNT parameters.

`DJDECOL=n`

Where n is from 1 to the record length of the input file. For example, `DJDECOL=3`.

DJDETRIG

This 1 - 10 character string is the DJDE identifier used in the DJDE records. This parameter is used in conjunction with the DJDECOL and DJDECNT parameters.

`DJDETRIG=string`

For example, `DJDETRIG=$DJDE`.

FIELD

Identifies the location of index data and can provide constant index values. You must define at least one field. You can define up to 128 fields.

The 390 indexer 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 can be used to index input data that contains one or more columns of sorted data. It is not practical to store every value in the database. The 390 indexer extracts the beginning and ending sorted values in each group.

Trigger FIELD syntax

```
FIELDn=record,column,length,(TRIGGER=n,BASE={0|TRIGGER})
```

Options and values

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

record
The relative record number from the trigger on which the field is based. This is the record number where the 390 indexer 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 the 390 indexer begins to search for the field. A value of 1 refers to the first byte in the record where the carriage control characters reside. 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. The column value can be 1 to 32756.

length
The number of contiguous bytes (characters) that compose the field. The supported range of values are 1 to 254.

TRIGGER=n
Identifies the trigger parameter that the 390 indexer uses to locate the field. Replace n with the number of a defined TRIGGER parameter.

BASE={0|TRIGGER}
Determines whether the 390 indexer uses the starting column number of the trigger string value to locate the field data. Choose from 0 or TRIGGER. If BASE=0, the indexer adds zero to the field column offset. If BASE=TRIGGER, the indexer 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 base 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.

Example

The following field parameter causes the 390 indexer to locate field values that begin in column 83 of the same record that contains the TRIGGER1 string value. The field length is eight bytes. We specify BASE=0 because the field data always starts in the same column.

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

The following field parameter causes the indexer 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 the indexer 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

```
FIELDn=constant
```

Options and values

n

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

constant

The literal (constant) string value of the field. This is the index value that gets stored in the database. The constant value can be 1 to 254 bytes in length. The 390 indexer does not perform any validity checking on the actual content of the supplied data.

Example

The following field parameter causes the 390 indexer to store the same string of hexadecimal characters in each INDEX4 value it creates.

```
FIELD3=X'F0F560F1F760F0F5'  /* CONSTANT 05/17/05 */
INDEX4=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD3,(TYPE=GROUP,BREAK=NO)
/* POSTING_DATE */
```

Transaction FIELD syntax - for INDEXn with GROUPRANGE

```
FFIELDn=*,*,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.

*****

The record number where the 390 indexer begins searching for the field. A transaction field must specify an asterisk, meaning the 390 indexer searches every record in the group.

*****

The column number where the 390 indexer begins searching for the field. A transaction field must specify an asterisk. The OFFSET specification determines the column or columns where the 390 indexer locates the field.

length

The number of contiguous bytes (characters) that compose the field. The supported range of values are 1 to 254.

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. You can use a maximum of eight pairs of beginning and ending offset values. 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, otherwise, the 390 indexer does not detect a match.

MASK='@#=^%'

Determines the pattern of symbols that the 390 indexer 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 indexes. Valid mask symbols include the following:

@

Matches alphabetic characters.

#

Matches numeric characters.

^

Matches any non-blank character.

%

Matches the blank character and numeric characters.

=

Matches any character.

ORDER={BYROW | BYCOL}

Identifies where the 390 indexer can locate the smallest value and the largest value of a group of sorted values that are arranged in either rows or columns on the page. The default ORDER is BYROW.

For ORDER=BYROW, the 390 indexer extracts the first value in the first row and the last value in the last row that match the MASK. Data within a row orientation might appear as follows:

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

For ORDER=BYCOL, the 390 indexer extracts the first value in the first column and the last value in the last column that match the MASK. Data with a column orientation might appear as follows:

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

Example

The following field parameter causes the 390 indexer 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)
```

Transaction FIELD syntax - for INDEXn with GROUPRANGE2
The FIELD syntax used in conjunction with INDEXn parameters using the GROUPRANGE2 type use the same syntax as the Trigger FIELD syntax. Refer to the Trigger FIELD syntax from preceding information.

The MASK sub-parameter cannot be used on FIELD parameters when they are used by INDEXn parameters with the TYPE=GROUPRANGE2 sub-parameter.

FILEFORMAT

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

Important: This parameter is required when the **INDEXSTYLE** parameter is not set toAFP.

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

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 in the STREAM format typically come from a workstation operating system such as AIX or 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 environments. The following example specifies the **FILEFORMAT** parameter to process input data that contains two-character line delimiters:

```
FILEFORMAT=STREAM,(NEWLINE=X’0D0A’)
```

If STREAM is specified, the NEWLINE keyword is required.

RECORD
The input file has a variable record format in which the first two bytes of each line, called the record descriptor word (RDW), specify the length of the line. The records can be from 1 to 32767 bytes in length, plus the two bytes for the RDW.

RECORD,*n*
The input file has a fixed record format, where each record has a fixed length of *n* bytes. The value of *n* is a number from 1 to 32767 and specifies the fixed length of the entire record.

GROUPMAXPAGES

Determines the maximum number of pages to be put into a group. This allows the 390 indexer to logically segment a large report into groups of pages and create indexes for each group. You can specify a number from 1 and 9999.

For INDEXSTYLE values of PAGE, PDOC and NODX: The GROUPMAXPAGES defaults to 100 if no value is provided.

For INDEXSTYLE of DOC:

- There is no default for GROUPMAXPAGES. If GROUPMAXPAGES is not specified for INDEXSTYLE of DOC, all pages are grouped into one document until the BREAK=YES condition(s) are met.
- When the GROUPMAXPAGES value is reached, the current document is closed and a new document is started.
- When a new document is started because the GROUPMAXPAGES value is reached (as opposed to because an index with BREAK=YES is satisfied), the index values used for the new document come from:
  - Indexes with BREAK=YES retain the values from the previous document.
  - Indexes with BREAK=NO will start with the values from the previous document, but attempts are made to locate new values in the new document to replace the previous values.

The GROUPMAXPAGES parameter is ignored when the INDEXSTYLE is AFP or when the ANYEXIT parameter is used.

INDEX

Identifies the index name and the field(s) on which the index is based. You must define at least one index parameter.

You can define up to 128 index parameters.

Syntax

```
INDEXn=name,FIELDnn[,...FIELDnn],TYPE=type
```

Options and values

n
The index parameter identifier. When you add an index parameter, use the next available number beginning with 1.

name
Determines the index name that is associated with the index value. The index name can be a maximum of 250 bytes. It is recommended that you enter the name as a hexadecimal string.

FIELDnn
Name of the field parameters that the 390 indexer uses to locate the index.

TYPE=*type*
You can specify *type* as follows:

```
TYPE=GROUP,BREAK={YES[,INITVAL=value] | NO} [,KEEP={YES | NO}]
TYPE=GROUPRANGE,BREAK=NO
TYPE=GROUPRANGE2,BREAK=NO
```

Ensure that you enter *value* as a hexadecimal string.

Notes

- TYPE=GROUP must be specified for all indexes except:
  - INDEX1 when INDEXSTYLE=PAGE
  - INDEX2 when INDEXSTYLE=PAGE or PDOC
  - INDEX3 when INDEXSTYLE=NODX
- TYPE=GROUPRANGE or TYPE=GROUPRANGE2 must be specified for:
  - INDEX1 when INDEXSTYLE=PAGE
  - INDEX2 when INDEXSTYLE=PDOC
- TYPE=GROUPRANGE must be specified for:
  - INDEX2 when INDEXSTYLE=PAGE
  - INDEX3 when INDEXSTYLE=NODX
- You can specify up to 32 FIELD parameters when you specify TYPE=GROUP. Separate field parameter names with a comma. When you specify multiple FIELD parameters on a single INDEXn parameter, each FIELD value is concatenated to form the index value. The total length of all the specified FIELD parameters cannot exceed 254 bytes. The FIELD values are ignored for indexes 1, 2, and 3 when you specify INDEXSTYLE=NODX.
- Only one FIELD value can be specified when TYPE=GROUPRANGE is specified.
- One or two FIELD values can be specified when TYPE=GROUPRANGE2 is specified.
- When TYPE=GROUPRANGE is used for:
  - INDEX1 when INDEXSTYLE=PAGE

– INDEX2 when INDEXSTYLE=PDOC

The single FIELD subparameter must refer to a FIELD parameter that uses the "mask" format, for example:

```
INDEX1=X 'D3D6C1D56DD5E4D4C2C5D9C',FIELD1,(TYPE=GROUPRANGE)
FIELD1=*,*,10,(OFFSET=(3:12),MASK= '##########' ,ORDER=BYROW)
```

- When TYPE=GROUPRANGE2 is used for:
  – INDEX1 when INDEXSTYLE=PAGE
  – INDEX2 when INDEXSTYLE=PDOC

  One or two FIELD subparameters can be specified. The FIELD parameter(s) pointed to cannot use the "mask" format.

  When one FIELD subparameter is specified, it refers to both the begin range value and the end range value.

```
INDEX1=X'D3D6C1D56DD5E4D4C2C5D9',FIELD1,(TYPE=GROUPRANGE2)
FIELD1=0,3,10,(TRIGGER=3,BASE=0)
TRIGGER3=*,46,X'4B',(TYPE=FLOAT)  /* . */
```

  This set of parameters is interpreted as the following:

  – **For the begin range value:** Start at the top of the page and scan down looking for a hexadecimal 4B in column 46 (per TRIGGER3). If found, then (per FIELD1=0,3,10) stay on this line (0), go to column 3 (3), and extract 10 positions (10).
  – **For the end range value:** Start at the bottom of the page and scan up looking for a hexadecimal 4B in column 46. If found, then stay on this line, go to column 3 and extract 10 positions. For

    ```
    FIELDn=x,y,z
    ```

    x is the number of lines to move from the line where the trigger value was found. If 0, stay on this line. If negative, move up that number of lines. If positive, move down that number of lines. y is the column number to start from on the offset line. z is the number of positions to extract, starting with column y.

  When two FIELD subparameters are specified, the first refers to the begin range value and the second refers to the end range value.

- When you specify TYPE=GROUP, use BREAK=YES for index values that must change to indicate that the current page is the start of a new document. However, if the index value contains multiple fields which are to be concatenated together, only the first field is examined to determine if the current page is the start of a new document.

  When you specify BREAK=YES, you can optionally specify the INITVAL=*value* parameter to give an initial value that the first field in the index value must change from before the first document is captured. This is one method to skip over alignment pages at the start of a print stream.

  The following is an example. Note that this should be entered all on one line.

```
INDEX3=x'C1C3C3D8E4D5E3',FIELD2,(TYPE=GROUP,BREAK=YES,
INITVAL=x'5C5C5C5C5C5C5C5C5C')
```

  In this example, all pages at the start of the report file that contain a string of asterisks in the position of the ACCOUNT field are skipped over. The report starts capturing its first document when something other than a string of asterisks is found in that position.

- When you specify TYPE=GROUP, you can optionally specify the KEEP={YES | NO} parameter. This parameter can be used with any TYPE=GROUP index.

  When KEEP=NO is used, this is the same as not specifying the KEEP parameter at all. Each document in the load file must provide all of its own index values.

When KEEP=YES is used, the index value found for the first document in the load file is used for that index across all remaining documents in the load file. Two examples of when this might be useful are:

  - The posting date field exists only in the first document of the load file.
  - An AFP file generates multiple index rows per document, but the posting date TLE does not repeat for each generated index row.

- Refer to the "INDEXSTYLE" on page 158 parameter for examples.

INDEXSTARTBY

The **INDEXSTARTBY** parameter determines the page number by which the 390 indexer must find a page where the **TRIGGER** parameters with **TYPE**=**GROUP** are found.

The 390 indexer fails if it does not find these values before the specified page number. This parameter is optional. It defaults to zero, which means that these **TRIGGER** parameters can be first found on any page. The maximum value for **INDEXSTARTBY** parameter 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, for example, **INDEXSTARTBY**=3, so that the 390 indexer does not start indexing until the page after the header pages.

Important: All pages found before the page containing the **TRIGGER** values, for **TRIGGER** parameters with **TYPE**=**GROUP**, are discarded.

Syntax

INDEXSTARTBY=value

Options and values

The value is the page number of the report by which the 390 indexer must find the **TRIGGER** values for **TRIGGER** parameters with **TYPE**=**GROUP**. A value of 0 indicates that there is no limit to the page where the 390 indexer must find the **TRIGGER** values.

INDEXSTYLE

This parameter defines the type of report being captured. Certain rules must be followed regarding the indexes defined in the Content Manager OnDemand application group to which the Content Manager OnDemand application is associated. If this parameter is not specified, the default of **INDEXSTYLE**=DOC is assumed.

The **INDEXSTYLE** parameter is ignored when the **ANYEXIT** parameter is specified.

All application groups should have an index defined on Posting Date, with a data type of DATE. This index should be marked as being the segment field for the application group.

Parameters

You can use the following valid values.

DOC

DOC reports are traditional document reports, such as statements, invoices, and so forth. No indexes of type GROUPRANGE or GROUPRANGE2 can be specified.

The following example shows typical indexing parameters and values for DOC reports. Indexing parameters are specified on the Indexer Information page in the Content Manager OnDemand application.

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,133
TRC=YES
TRIGGER1=*,1,X'F1',(TYPE=GROUP)  /* 1 */
TRIGGER2=*,3,X'D7C1C7C540F140',(TYPE=GROUP)  /* PAGE 1  */
TRIGGER3=*,73,X'C1C3C3D6E4D5E3',(TYPE=FLOAT)  /* ACCOUNT */
TRIGGER4=*,3,X'C3D6D5E3C5D5E3E2',(TYPE=FLOAT)  /* CONTENTS */
TRIGGER5=*,19,X'C5D5C4C9D5C740C2C1D3',(TYPE=FLOAT)  /* ENDING BAL */
```

```
FIELD1=-1,89,9,(TRIGGER=3,BASE=0)
FIELD2=1,87,11,(TRIGGER=4,BASE=0)
FIELD3=7,19,12,(TRIGGER=1,BASE=0)
FIELD4=0,87,16,(TRIGGER=5,BASE=0)
FIELD5=0,37,8,(TRIGGER=5,BASE=0)
INDEX1=X'C1C3C3D6E4D5E36DD5E4D4C2C5D9',FIELD1,(TYPE=GROUP,BREAK=YES)  /* ACCOUNT_NUMBER */
INDEX2=X'E2E2D56D6D6DE3C1E76DC9C4',FIELD2,(TYPE=GROUP,BREAK=NO)  /* SSN___TAX_ID */
INDEX3=X'C3E4E2E36DD5C1D4C5',FIELD3,(TYPE=GROUP,BREAK=NO)  /* CUST_NAME */
INDEX4=X'C5D5C4C9D5C76DC2C1D3C1D5C3C5',FIELD4,(TYPE=GROUP,BREAK=NO)  /* ENDING_BALANCE */
INDEX5=X'C3D3D6E2C56DC4C1E3C5',FIELD5,(TYPE=GROUP,BREAK=NO)  /* CLOSE_DATE */
INDEX6=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD5,(TYPE=GROUP,BREAK=NO)  /* POSTING_DATE */
INDEXSTYLE=DOC
```

PAGE

PAGE reports are *transaction* type reports. The entire report is sorted by some column value. This sort key is used in the first and second indexes. The **GROUPMAXPAGES** parameter can be used to determine the number of pages included in each segment. If no **GROUPMAXPAGES** value is specified, the default is 100 pages.

The indexes defined for the Content Manager OnDemand application group must be as follows:

First Index
Must be the start value for a GROUPRANGE or GROUPRANGE2 index.

Second Index
Must be the end value for the GROUPRANGE or GROUPRANGE2 index.

Third Index
Must be the start value for a GROUPRANGE index on Page Number and must be defined as an integer. The value for the third index is set by the 390 indexer by counting the pages as they are stored, not from values extracted from the report data.

Fourth Index
Must be the end value for the GROUPRANGE index on Page Number and must be defined as an integer. The value for the fourth index is set by the 390 indexer by counting the pages as they are stored, not from values extracted from the report data.

Additional indexes
Can be defined, but cannot be GROUPRANGE or GROUPRANGE2 indexes.

The following parameter list shows the typical indexing parameters and values for PAGE reports using TYPE=GROUPRANGE on INDEX1. Indexing parameters are specified on the **Indexer Information** page in the Content Manager OnDemand application.

The mask subparameter of the **FIELD1** parameter in the following list of parameters instructs the indexer to look in columns 3 through 12 for a value that matches the mask. The first value found in a document is used as the start range value and stored in the first index field of the Application Group Data table. The last value found in a document is used as the end range value and stored in the second index field of the Application Group Data table.

The following example shows typical indexing parameters and values for PAGE reports using TYPE=GROUPRANGE for INDEX1:

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,90
GROUPMAXPAGES=100
TRIGGER1=*,1,X'F1',(TYPE=GROUP)  /* 1 */
TRIGGER2=0,2,X'D9C5D7D6D9E3',(TYPE=GROUP)  /* REPORT */
FIELD1=*,*,10,(OFFSET=(3:12),MASK='##########',ORDER=BYROW)
FIELD2=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'D3D6C1D56DD5E4D4C2C5D9',FIELD1,(TYPE=GROUPRANGE)  /* LOAN_NUMBER */
INDEX2=X'D7C1C7C56DD5D66D',FIELD1,(TYPE=GROUPRANGE)  /* PAGE_NO_ */
INDEX3=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD2,(TYPE=GROUP,BREAK=NO)  /* POSTING_DATE */
INDEXSTYLE=PAGE
```

This parameter list shows typical indexing parameters and values for PAGE reports using TYPE=GROUPRANGE2 on INDEX1. Indexing parameters are specified on the **Indexer Information** page in the Content Manager OnDemand application.

In this example, FIELD1 is used to locate the start range value. FIELD1 uses TRIGGER1. In this case, when a skip-to-channel-1 is found, then go down eight lines (to line number 9 of this page) and extract ten positions starting at column 3.

FIELD2 is used to locate the end range value. FIELD2 uses TRIGGER3. In this case, start at the bottom of the page and look up in column 46 for a period (x'4B'). When found, stay on that line and extract ten positions starting at column 3.

The following example shows typical indexing parameters and values for PAGE reports using TYPE=GROUPRANGE2 for INDEX1:

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,90
GROUPMAXPAGES=100
TRIGGER1=*,1,X'F1',(TYPE=GROUP) /* 1 */
TRIGGER2=0,2,X'D9C5D7D6D9E3',(TYPE=GROUP) /* REPORT */
TRIGGER3=*,46,X'4B',(TYPE=FLOAT)  /* . */
FIELD1=8,3,10,(TRIGGER=1,BASE=0)
FIELD2=0,3,10,(TRIGGER=3,BASE=0)
FIELD3=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'D3D6C1D56DD5E4D4C2C5D9',FIELD1,FIELD2,(TYPE=GROUPRANGE2) /* LOAN_NUMBER */
INDEX2=X'D7C1C7C56DD5D66D',FIELD1,(TYPE=GROUPRANGE) /* PAGE_NO_ */
INDEX3=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD3,(TYPE=GROUP,BREAK=NO) /* POSTING_DATE */
INDEXSTYLE=PAGE
```

The following parameter list shows another set of typical indexing parameters and values for PAGE reports using TYPE=GROUPRANGE2 on INDEX1. Indexing parameters are specified on the Indexer Information page in the Content Manager OnDemand application.

In this example, FIELD1 is used to locate both the start and end range values. FILED1 uses TRIGGER3. For the start range value, start at the top of the page and look down in column 46 for a period (x'4B'). When found, stay on that line and extract ten positions starting at column 3.

For the end range value, start at the bottom of the page and look up in column 46 for a period (x'4B'). When found, stay on that line and extract ten positions starting at column 3.

The following example shows typical indexing parameters and values for PAGE reports using TYPE=GROUPRANGE2 for INDEX1:

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,90
GROUPMAXPAGES=100
TRIGGER1=*,1,X'F1',(TYPE=GROUP) /* 1 */
TRIGGER2=0,2,X'D9C5D7D6D9E3',(TYPE=GROUP) /* REPORT */
TRIGGER3=*,46,X'4B',(TYPE=FLOAT)  /* . */
FIELD1=0,3,10,(TRIGGER=3,BASE=0)
FIELD2=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'D3D6C1D56DD5E4D4C2C5D9',FIELD1,(TYPE=GROUPRANGE2) /* LOAN_NUMBER */
INDEX2=X'D7C1C7C56DD5D66D',FIELD1,(TYPE=GROUPRANGE) /* PAGE_NO_ */
INDEX3=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD2,(TYPE=GROUP,BREAK=NO) /* POSTING_DATE */
INDEXSTYLE=PAGE
```

Note the following items:

- The FIELDn value pointed to by the INDEX2 Page Number index is needed to meet the syntax checking requirements of the graphical indexer, but is not used by the 390 indexer. Any valid FIELDn value may be specified for INDEX2.
- The GROUPRANGE specification for INDEX2 causes the beginning page number value to be stored in the third application group index field, while the ending page number value gets stored in the fourth application group index field. INDEX3 (posting date from preceding information) gets stored in the fifth application group index field.

PDOC

PDOC reports are transaction type reports, but have a high level index. For example, a bank might have a report that is organized by Branch Number. Within each branch, the report is sorted on some column. The **GROUPMAXPAGES** parameter can be used to determine the number of pages included in each document.

If no **GROUPMAXPAGES** value is specified, the default is 100 (pages). A new document is started when either the high level index changes or the **GROUPMAXPAGES** value is reached.

The indexes defined for the Content Manager OnDemand application group must be as follows:

First Index
Must be the high level index.

Second Index
Must be the start value for the GROUPRANGE or GROUPRANGE2 index by which the report is sorted within the first index.

Third Index
Must be the end value for the GROUPRANGE or GROUPRANGE2 index by which the report is sorted within the first index.

Additional indexes
Might be defined, but cannot be GROUPRANGE or GROUPRANGE2 indexes.

The following example lists typical indexing parameters and values for PDOC reports using **TYPE=GROUPRANGE** on INDEX2. Indexing parameters are specified on the Indexer Information page in the Content Manager OnDemand application.

The mask subparameter of the **FIELD2** parameter in the following example instructs the indexer to look in columns 3 through 12 for a value that matches the mask. The first value found in a document is used as the start range value and stored in the first index field of the Application Group Data table. The last value found in a document is used as the end range value and stored in the second index field of the Application Group Data table.

The following example shows typical indexing parameters and values for PDOC reports that use **TYPE=GROUPRANGE** for INDEX2:

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,90
GROUPMAXPAGES=100
TRIGGER1=*,1,X'F1',(TYPE=GROUP)  /* 1 */
TRIGGER2=*,2,X'C2C1D5D2',(TYPE=GROUP)  /* BANK */
FIELD1=1,11,3,(TRIGGER=1,BASE=0)
FIELD2=*,*,10,(OFFSET=(3:12),MASK='##########',ORDER=BYROW)
FIELD3=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'C2C1D5D26DC2D9C1D5C3C8',FIELD1,(TYPE=GROUP,BREAK=YES)  /* BANK_BRANCH */
INDEX2=X'D3D6C1D56DD5E4D4C2C5D9',FIELD2,(TYPE=GROUPRANGE)  /* LOAN_NUMBER */
INDEX3=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD3,(TYPE=GROUP,BREAK=NO)  /* POSTING_DATE */
INDEXSTYLE=PDOC
```

The following example lists typical indexing parameters and values for PDOC reports using **TYPE=GROUPRANGE2** on INDEX2. Indexing parameters are specified on the **Indexer Information** page in the Content Manager OnDemand application.

In this example, FIELD2 is used to locate the start range value. FIELD2 uses **TRIGGER1**. In this case, when a skip-to-channel-1 is found, then go down eight lines (to line number 9 of this page) and extract ten positions starting at column 3.

FIELD3 is used to locate the end range value. FIELD3 uses **TRIGGER3**. In this case, start at the bottom of the page and look up in column 46 for a period (x'4B'). When found, stay on that line and extract ten positions starting at column 3.

The following example shows typical indexing parameters and values for PDOC reports using **TYPE=GROUPRANGE2** for INDEX2:

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,90
GROUPMAXPAGES=100
TRIGGER1=*,1,X'F1',(TYPE=GROUP) /* 1 */
TRIGGER2=*,2,X'C2C1D5D2',(TYPE=GROUP) /* BANK */
TRIGGER3=*,46,X'4B',(TYPE=FLOAT)  /* . */
FIELD1=1,11,3,(TRIGGER=1,BASE=0)
FIELD2=8,3,10,(TRIGGER=1,BASE=0)
FIELD3=0,3,10,(TRIGGER=3,BASE=0)
```

```
FIELD4=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'C2C1D5D26DC2D9C1D5C3C8',FIELD1,(TYPE=GROUP,BREAK=YES) /* BANK_BRANCH */
INDEX2=X'D3D6C1D56DD5E4D4C2C5D9',FIELD2,FIELD3,(TYPE=GROUPRANGE2) /* LOAN_NUMBER */
INDEX3=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD4,(TYPE=GROUP,BREAK=NO) /* POSTING_DATE */
INDEXSTYLE=PDOC
```

The following example lists another set of typical indexing parameters and values for PDOC reports using **TYPE=GROUPRANGE2** on **INDEX2**. Indexing parameters are specified on the **Indexer Information** page in the Content Manager OnDemand application.

In this example, FIELD2 is used to locate both the start and end range values. FILED2 uses **TRIGGER3**. For the start range value, start at the top of the page and look down in column 46 for a period (x'4B'). When found, stay on that line and extract ten positions starting at column 3.

For the end range value, start at the bottom of the page and look up in column 46 for a period (x'4B'). When found, stay on that line and extract ten positions starting at column 3.

The following example shows typical indexing parameters and values for PDOC reports using **TYPE=GROUPRANGE2** for INDEX2:

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,90
GROUPMAXPAGES=100
TRIGGER1=*,1,X'F1',(TYPE=GROUP) /* 1 */
TRIGGER2=*,2,X'C2C1D5D2',(TYPE=GROUP) /* BANK */
TRIGGER3=*,46,X'4B',(TYPE=FLOAT)  /* . */
FIELD1=1,11,3,(TRIGGER=1,BASE=0)
FIELD2=0,3,10,(TRIGGER=3,BASE=0)
FIELD3=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'C2C1D5D26DC2D9C1D5C3C8',FIELD1,(TYPE=GROUP,BREAK=YES) /* BANK_BRANCH */
INDEX2=X'D3D6C1D56DD5E4D4C2C5D9',FIELD2,(TYPE=GROUPRANGE2) /* LOAN_NUMBER */
INDEX3=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD3,(TYPE=GROUP,BREAK=NO) /* POSTING_DATE */
INDEXSTYLE=PDOC
```

The **GROUPRANGE** or **GROUPRANGE2** specification for INDEX2 will cause the beginning loan number value to be stored in the second application group index field, while the ending loan number value gets stored in the third application group index field. INDEX3 (posting date from preceding information) gets stored in the fourth application group index field.

NODX

NODX (no index) reports are ones which either do not have obvious index values, or which are very short and do not need to be broken up into documents. The **GROUPMAXPAGES** parameter can be used to determine the number of pages included in each segment. If no **GROUPMAXPAGES** value is specified, the default is 100 (pages).

The indexes defined for the Content Manager OnDemand application group must be as follows:

First Index
: Must be defined as Segment Number. Must be defined as an integer. The 390 indexer assigns values to this index by sequentially counting each segment (document) as it is created.

Second Index
: Must be defined as Report Date. Must be defined as a string of length 8 (eight). The 390 indexer assigns this value, based on the Posting Date of the report. The value will have the format of MM/DD/YY. A separate index defined on Posting Date with a data type of DATE should also be defined. This index should be marked as being the segment field for the application group

Third Index
: Must be the start value for a GROUPRANGE index on Page Number. Must be defined as an integer. The value for the third index is set by the 390 indexer by counting the pages as they are stored, not from values extracted from the report data.

Fourth Index
: Must be the end value for the GROUPRANGE index on Page Number. Must be defined as an integer. The value for the fourth index is set by the 390 indexer by counting the pages as they are stored, not from values extracted from the report data.

Additional indexes
May be defined, but cannot be GROUPRANGE or GROUPRANGE2 indexes.

The following example lists typical indexing parameters and values for NODX reports. Indexing parameters are specified on the Indexer Information page in the Content Manager OnDemand application.

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,90
GROUPMAXPAGES=50
TRIGGER1=*,1,X'F1',(TYPE=GROUP)  /* 1 */
FIELD1=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'E2C5C7D4C5D5E36DD5E4D4C2C5D9',FIELD1,(TYPE=GROUP,BREAK=NO)  /* SEGMENT_NUMBER */
INDEX2=X'D9C5D7D6D9E36DC4C1E3C5',FIELD1,(TYPE=GROUP,BREAK=NO)  /* REPORT_DATE */
INDEX3=X'D7C1C7C56DD5E4D4C2C5D9',FIELD1,(TYPE=GROUPRANGE)  /* PAGE_NUMBER */
INDEX4=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD1,(TYPE=GROUP,BREAK=NO)  /* POSTING_DATE */
INDEXSTYLE=NODX
```

Remember:

- The FIELD*n* value pointed to by the INDEX1, INDEX2, and INDEX3 indexes are needed to meet the syntax checking requirements of the graphical indexer, but are not used by the 390 indexer. Any validFIELD*n* value may be specified for these indexes.
- The **GROUPRANGE** specification for INDEX3 will cause the beginning page number value to be stored in the third application group index field, while the ending page number value gets stored in the fourth application group index field. INDEX4 (posting date from preceding information) gets stored in the fifth application group index field.

AFP
AFP (Advanced Function Printing) reports captured through the 390 indexer must already have been formatted into an AFP Data Stream (AFPDS). This can be done by using ACIF (AFP Conversion and Indexing Facility) or by any third party program. The 390 indexer looks for index values within the AFPDS, either in TLE or NOP records. ACIF, and other programs, can automatically generate the TLE records. The NOP records for use by the 390 indexer have a fixed format.

The NOP record format provides space for 32 indexes of up to 256 characters each. One ODZOSSEG record must exist for each document. This identifies that a new document is starting, and provides one complete set of index values for the document.

One or more ODZOSDIR records can exist for each document. This record provides an additional set of index values for the document.

NOP record formats for earlier releases of Content Manager OnDemand for z/OS and R/DARS-ESA are supported by the 390 indexer for compatibility purposes.

The following table shows the layout of the ODZOSSEG and ODZOSDIR NOP record.

Table 11. Layout of the ODZOSSEG and ODZOSDIR NOP record

Position	Description
1	X'5A'
2 - 3	Length of this record - 1
4 - 6	X'D3EEEE'
7 - 9	X'000000'
10 - 17	'ODZOSSEG' or 'ODZOSDIR'
18 - 273	Value of field 1 (256 bytes)
274 - 529	Value of field 2 (256 bytes)
530 - 785	Value of field 3 (256 bytes)

Table 11. Layout of the ODZOSSEG and ODZOSDIR NOP record (continued)

Position	Description
786 - 1041	Value of field 4 (256 bytes)
1042 - 1297	Value of field 5 (256 bytes)
1298 - 1553	Value of field 6 (256 bytes)
1554 - 1809	Value of field 7 (256 bytes)
1810 - 2065	Value of field 8 (265 bytes)
2066 - 2321	Value of field 9 (256 bytes)
2322 - 2577	Value of field 10 (256 bytes)
2578 - 2833	Value of field 11 (256 bytes)
2834 - 3089	Value of field 12 (256 bytes)
3090 - 3345	Value of field 13 (256 bytes)
3346 - 3601	Value of field 14 (256 bytes)
3602 - 3857	Value of field 15 (256 bytes)
3858 - 4113	Value of field 16 (256 bytes)
4114 - 4369	Value of field 17 (256 bytes)
4370 - 4625	Value of field 18 (256 bytes)
4626 - 4881	Value of field 19 (256 bytes)
4882 - 5137	Value of field 20 (256 bytes)
5138 - 5393	Value of field 21 (256 bytes)
5394 - 5649	Value of field 22 (256 bytes)
5650 - 5905	Value of field 23 (256 bytes)
5906 - 6161	Value of field 24 (256 bytes)
6162 - 6417	Value of field 25 (256 bytes)
6418 - 6673	Value of field 26 (256 bytes)
6674 - 6929	Value of field 27 (256 bytes)
6930 - 7185	Value of field 28 (256 bytes)
7186 - 7441	Value of field 29 (256 bytes)
7442 - 7697	Value of field 30 (256 bytes)
7698 - 7953	Value of field 31 (256 bytes)
7954 - 8209	Value of field 32 (256 bytes)

The following example lists typical indexing parameters and values for AFP reports that use TLE records. Indexing parameters are specified on the Indexer Information page in the Content Manager OnDemand application.

```
TRIGGER1=*,1,X'5A',(TYPE=GROUP)  /* AFP x'5A' */
FIELD1=-0,1,14,(TRIGGER=1,BASE=0)
FIELD2=0,1,24,(TRIGGER=1,BASE=0)
FIELD3=0,1,18,(TRIGGER=1,BASE=0)
INDEX1=X'D796938983A8',FIELD1,(TYPE=GROUP,BREAK=YES)  /* Policy */
```

```
INDEX2=X'C39695A38595A3A2',FIELD2,(TYPE=GROUP,BREAK=NO)  /* Contents */
INDEX3=X'C995A2A4998584',FIELD3,(TYPE=GROUP,BREAK=NO) /* Insured */
INDEXSTYLE=AFP
```

In the previous example, the TRIGGER record is not used by the 390 indexer and is specified only to meet syntax checking requirements of the graphical indexer. The only value used from the FIELD records is the length value. The name fields of the INDEX values must match the Attribute Name field of the TLE records, and is used to map the index values back to the Application Group data table columns. The **BREAK** parameter of the INDEX record is not used.

The following example lists typical indexing parameters and values for AFP reports using NOP records. Indexing parameters are specified on the Indexer Information page in the Content Manager OnDemand application.

```
TRIGGER1=*,1,X'5A',(TYPE=GROUP)  /* AFP x'5A' */
FIELD1=-0,1,15,(TRIGGER=1,BASE=0) /* Length of data to extract = 15 */
FIELD2=0,1,11,(TRIGGER=1,BASE=0)  /* Length of data to extract = 11 */
FIELD3=0,1,8,(TRIGGER=1,BASE=0)   /* Length of data to extract = 8  */
FIELD4=0,1,8,(TRIGGER=1,BASE=0)   /* Length of data to extract = 8  */
INDEX1=X'F1',FIELD1,(TYPE=GROUP,BREAK=NO)   /* CUST_NAME    in NOP Field 1 */
INDEX2=X'F2',FIELD2,(TYPE=GROUP,BREAK=YES)  /* ACCOUNT_NUM  in NOP Field 2 */
INDEX3=X'F5',FIELD3,(TYPE=GROUP,BREAK=NO)   /* REPORT_DATE  in NOP Field 5 */
INDEX4=X'F6',FIELD4,(TYPE=GROUP,BREAK=NO)   /* POSTING_DATE in NOP Field 6 */
INDEXSTYLE=AFP
```

In the previous example, the TRIGGER record is not used by the 390 indexer and is specified only to meet syntax checking requirements of the graphical indexer. The only value used from the FIELD records is the length value. The name fields of the INDEX values are character representations of numbers which point to the position within the NOP record where the index value is to be found for each index. The **BREAK** parameter of the INDEX record is not used.

Special consideration is needed when dealing with the Posting Date field with the older style of NOP records.

- Content Manager OnDemand for z/OS V2.1 used NOP record types of OD390SEG and OD390DIR. The INDEX record for the Posting Date field must specify a name value of X'F1F7', for example:

  ```
  INDEX4=X'F1F7',FIELD4,(TYPE=GROUP,BREAK=NO) /*POSTING_DATE in OD390SEG NOP*/
  ```

- Content Manager OnDemand for z/OS V1.1 and R/DARS-ESA used NOP record types of RDARSSEG and RDARSDIR. The INDEX record for the Posting Date field must specify a name value of X'F6', for example:

  ```
  INDEX4=X'F6',FIELD4,(TYPE=GROUP,BREAK=NO) /*POSTING_DATE in RDARSSEG NOP*/
  ```

INDXEXIT

The Index Exit is provided to allow the report indexes to be modified prior to insertion into the Application Group Data Table. This exit can be used with any type of report captured by the 390 indexer. The exit is called dynamically during the capture process. The capture program calls the exit when the indexing instructions for the application include the INDXEXIT parameter. The report administrator provides a program name for the Index Exit.

There are no restrictions as to the type of processing that can be performed in an index exit with the exception that the exit must pass the standard parameter list back to the capture program. A sample COBOL exit is provided in ARSEXNDX, along with the COBOL copybook ARSINDBK. A sample C exit is provided in ARSECNDX along with the C header file ARSINDBH.

Parameters

The following example from the **ARSINDBK** Cobol Copybook shows the parameters that are required by **ARSEXNDX**.

```
01  WS-HEADER-RECORD.
    05 FIRST-FIVE                                  PIC X(5).
    05 WS-KEY-IND                                  PIC X(3).
```

```
    05 WS-FIELDS.
    10 WS-FIELD OCCURS 32 TIMES                          PIC X(256).
    05 WS-APPLICATION-NAME                               PIC X(60).
    05 WS-IDX-FOUND-SET                                  PIC X.
    88 WS-IDX-FOUND-SET-YES                              VALUE 'Y'.
     88 WS-IDX-FOUND-SET-NO                              VALUE 'N'.
    05 WS-IDX-FOUND-TABLE.
       10 WS-F-FIELD OCCURS 32 TIMES.
          15 WS-F-FIELD-FOUND                            PIC X.
             88 WS-F-FIELD-FOUND-YES                     VALUE 'Y'.
             88 WS-F-FIELD-FOUND-NO                      VALUE 'N'.

01  WS-INDEX-NAME-COUNT-ORIGINAL                         PIC S9(8) COMP.
01  WS-INDEX-NAMES.
    05 WS-INDEX-NAME-COUNT-UPDATED                       PIC S9(8) COMP.
    05 WS-INDEX-NAME-TABLE OCCURS 32 TIMES.
       10 WS-INDEX-NAME                                  X(250).
```

FIRST-FIVE
Set by capture program. `Control` field.

WS-KEY-IND
Set by capture program. `Control` field.

WS-FIELD
Set by capture program. The value for the 32 Keys extracted from the report input. The exit can return modified values.

WS-APPLICATION-NAME
Set by capture program. The value of the Application Name for this report.

WS-IDX-FOUND-SET
Set by both capture program and exit. **WS-IDX-FOUND-SET-NO** is set on by the capture program prior to calling this exit.

WS-F-FIELD-FOUND
Set by capture program. **WS-F-FIELD-YES** is set on by the capture program for index values already found.

WS-INDEX-NAME-COUNT-ORIGINAL
Set by capture program. The number of indexes identified by the **INDEX** indexing parameters.

WS-INDEX-NAME-COUNT-UPDATED
Set by both capture program and exit. If the exit adds additional index field names not present in the **INDEX** indexing parameters, this count must be updated by the exit to indicate the new total number of indexes.

WS-INDEX-NAME
Set by both capture program and exit. Contains the name field from the **INDEX** indexing parameters. The exit can interrogate this array to determine which position in the **WS-FIELD** array represents which index value. If the exit adds additional index field names not present in the **INDEX** indexing parameters, this array must be updated to give the field name. This must match the Application Group field name.

Notes

All parameter fields are set before the Index Exit is called. The exit can alter the **WS-FIELD** array.

- **WS-IDX-FOUND-SET-NO** is set on by the calling program.
- The **WS-F-FIELD-FOUND-YES** flag is set on prior to this exit being called for all indexes already found.
- After the Index Exit returns control, if **WS-IDX-FOUND-SET-NO** is still on, then the calling routine will act as though the **WS-F-FIELD-FOUND-YES** flag is on for each valid index field. This will allow the process to work as it did before this change being added to V8.4.2, so existing index exits do not need to be modified.
- If the exit turns on the **WS-IDX-FOUND-SET-YES FLAG**, then the **WS-F-FIELD-FOUND-YES** flag must be set on for each index being returned. This is important for any index fields set by the Index Exit for which values were not already found.

- **WS-INDEX-NAME-COUNT-ORIGINAL** and **WS-INDEX-NAME-COUNT-UPDATED** are both set before each time the index exit is called. They are set to the number of **INDEX** indexing parameters present in the application definition.
- If the exit is adding an index value to the **WS-FIELD** array for an application group field for which there is no **INDEX** indexing parameter, then the **WS-INDEX-NAME-COUNT-UPDATED** field must be set to the new total number of indexes. The name of the index field, as defined in the application group, must be added to the **WS-INDEX-NAME** array.

The sample exit routine performs the following: receives the parameter list, performs report unique index modifications and returns control to the capture process. All processing is controlled by the MAINLINE section of the exit. Error messages are written to file INDSTATS.

The sample code performed in paragraph REPORT-UNIQUE-LOGIC is an example of the kind of processing that may be performed in an index exit. The REPORT-UNIQUE-LOGIC routine must be customized or removed for the exit to function properly with your report.

Developing an index exit

Procedure

The following approach is recommended for developing an index exit:

1. Create report definition with an index exit name specified in the INDXEXIT parameter.
2. Create a new member in the Sample library corresponding to the index exit name.
3. Copy the sample exit into the new member.
4. Modify exit as required.
5. Compile and link the exit.
6. Test.

INPEXIT

The Input Exit is provided to allow additional processing of the report input before the report is stored. This exit can only be used when the INDEXSTYLE is not set to AFP and when the ANYEXIT is not specified.

The exit processes two files. The report input file, called OBJINPT in the sample exit, and an error message file, called INPSTATS.

The sample exit routine performs the following: open the input file, read the input file, build the page buffer, perform report unique processing, and return control to the report capture program. All processing is controlled by the MAINLINE section of the exit. Changes related to file control processing may be required.

Important: Care should be taken when changing the routines that build the page buffer. Do not alter the size of the INPUT-DATALINE.

The sample code found in paragraph REPORT-UNIQUE-LOGIC is an example of the kind of processing that can be performed in an input exit.

Important: The REPORT-UNIQUE-LOGIC routine must be customized or removed for the exit to function properly with your report input.

The exit is called dynamically during the report capture process. The report capture routine calls the exit when the indexing parameters specify an input exit name in the **INPEXIT** parameter. The report administrator provides a program name for this parameter.

There are no restrictions as to the type of processing that can be performed in an input exit with the exception that the exit must pass the standard parameter list back to the report capture program. Values must be supplied for all parameters.

Beginning with Content Manager OnDemand for z/OS, Version 8.4 or later, a line print file can have a fixed record length greater than 512 or a variable record length. To support this capability, a new parameter format is provided. The old parameter format is still supported for backwards compatibility. The parameter format that is used when an input exit is called is determined as follows:

Table 12. How Content Manager OnDemand determines whether to use the old parameter format or the new parameter format

InputRecord Format	InputRecord Length	INPEXITNEW parameter value	z/OS or AIX	Parameter format used
Variable	all	Ignored	both	New
Fixed	512 or less	N or not specified	z/OS	Old
Fixed	512 or less	Ignored	AIX	New
Fixed	More than 512	Y	both	New
Fixed	More than 512	Ignored	both	New

Old parameter format
A sample exit is provided in ARSEXINP.

Parameters

The following example from the ARSINPBK Cobol Copybook shows the parameters that are required by ARSEXINP.

```
01 PAGE-BUFFER.
   02 PAGE-BUFFER-LINE OCCURS 256 TIMES.
      05 PAGE-BUFFER-CHAR PIC X(512).
01 LINE-COUNT COMP PIC S9(8).
01 RECORD-STATUS PIC X(3).
   88 END-OF-FILE VALUE 'EOF '.
01 ARSEXINP-DDNAME PIC X(8).
```

PAGE-BUFFER
Set by Exit. One data page of the report input. The buffer dimensions are: 512 characters wide and 256 lines per page. The dimensions must not be changed.

LINE-COUNT
Set by Exit. The number of lines in the Page Buffer.

RECORD-STATUS
Set by Exit. The end of file (EOF) indicator.

ARSEXINP-DDNAME
Set by calling routine. The DD Name of the input file. This is particularly useful when the load process is initiated by the Spool Capture process, because the DD Name will be different each time.

New parameter format

A sample COBOL exit is provided in ARSE2INP along with the COBOL copybook ARSIN2BK. A sample C exit is provided in ARSECINP along with the C header file ARSZ390H.

Parameters

The following example from the ARSIN2BK Cobol Copybook shows the parameters that are required by ARSE2INP.

```
01 PAGE-BUFFER.
   02 PAGE-BUFFER-AREA     OCCURS 1048320 PIC X.
01 LINE-COUNT                                    COMP PIC S9(8).
01 RECORD-STATUS                                        PIC X(3).
   88 END-OF-FILE                                    VALUE 'EOF'.
01 ARSE2INP-DDNAME                                    PIC X(8).
01 ARSE2INP-FILE-NAME PIC X(1023).

01 ARSE2INP-OS-LEVEL-FLAG PIC X.
   88 ARSE2INP-OS-LEVEL-ZOS VALUE 'Z'.
   88 ARSE2INP-OS-LEVEL-MP VALUE 'M'.
```

PAGE-BUFFER
Set by Exit. One data page of the report input. The records are concatenated into this buffer area. Each record must be preceded by a two byte value representing the record length. This two byte value does not include the length of itself.

If the PAGE-BUFFER-AREA is not large enough to hold an entire page, consider using the ANYSTORE exit instead of the INPUT exit.

LINE-COUNT
Set by Exit. The number of lines in the Page Buffer.

RECORD-STATUS
Set by Exit. The end of file (EOF) indicator.

ARSE2INP-DDNAME
Set by calling routine. The DD Name of the input file. This is particularly useful when the load process is initiated by the Spool Capture process because the DD Name will be different each time.

Developing an input exit

Procedure

The following approach is recommended for developing an input exit:

1. Create report definition with an input exit name specified in the **INPEXIT** parameter.
2. Create a new member in the Sample library corresponding to the input exit name.
3. Copy the sample exit into the new member.
4. Modify the exit as required.
5. Compile and link the exit.
6. Test.

INPEXITNEW

This parameter works with the INPEXIT parameter. If the INPEXIT parameter is not specified, this parameter is ignored.

Because a line print file can have a variable record length or a fixed record length, the 390 indexer can use one of two parameter lists to call the input exit.

When the input record format is fixed and the input record length is less than or equal to 512, the old parameter format is used by default. If you want to use the new parameter format in this situation, specify `INPEXITNEW=Y`. In all other situations, the new parameter format will be used when calling the input exit. See Table 13 on page 169 for details on when each parameter format is used.

Table 13. Explanation of when the old parameter format and new parameter format are used

InputRecord Format	InputRecord Length	INPEXITNEW parameter value	z/OS or AIX	Parameter format used
Variable	all	ignored	both	New
Fixed	512 or less	N or not specified	z/OS	Old
Fixed	512 or less	ignored	AIX	New
Fixed	More than 512	Y	both	New
Fixed	More than 512	ignored	both	New

LINEOFFSET

The **LINEOFFSET** parameter specifies whether to take the carriage control values into account when determining the number of lines to move up or down from the line that contains the trigger value.

The **FIELD** indexing parameter is used to locate index values in the load file. It points to a **TRIGGER** parameter that defines a string value in the load file to be used as a reference point to locate the index value. The **FIELD** parameter includes a record value that specifies the offset from the trigger value to the record that contains the index value. This record (or offset) value counts the number of lines to move up or down, from the record that contains the trigger value to get to the line that contains the index value.

Specifying LINEOFFSET=ASREAD indicates that the carriage control is not to be considered when counting lines. The lines are counted as they appear in the load file.

Specifying LINEOFFSET=ASPRINTED indicates that the carriage control is to be considered while counting lines.

Only the 0 (space two lines) and the dash (space three lines) are supported. The + (overstrike) character is not supported. Any carriage control values other than those supported are treated, for the purpose of determining record offset values, the same as the "Space one line" action.

The following examples show the first three records of a load file.

```
1REPORT
-ACCOUNT 777777
0JOHN SMITH
```

Using **ASREAD** (the default), the indexing parameters to collect the account number and name would be as follows:

```
TRIGGER1=*,2,'REPORT',(TYPE=GROUP)
FIELD1=1,10,6,(TRIGGER=1,BASE=0)
FIELD2=2,2,10,(TRIGGER=1,BASE=0)
```

Using **ASPRINTED**, the indexing parameters to collect the account number and name would be as follows:

```
TRIGGER1=*,2,'REPORT',(TYPE=GROUP)
FIELD1=3,10,6,(TRIGGER=1,BASE=0)
FIELD2=5,2,10,(TRIGGER=1,BASE=0)
```

The **LINEOFFSET** parameter is ignored when INDEXSTYLE=AFP is specified or when an **ANYSTORE** exit is used.

Syntax

```
LINEOFFSET= value
```

Options and values

The values can be:

ASREAD
Carriage controls are not used to calculate the record offsets for the fields. The offsets are relative to the lines as they are read from the load file. This value is the default if the **LINEOFFSET** parameter is not specified or an invalid value is specified.

ASPRINTED
Carriage controls are used to calculate the record offsets for the fields. The offsets are relative to the line spacing that occurs when the lines are printed.

Notes for index usage

*Table 14. Supported carriage controls in use by **ASPRINTED***

Action	ANSI (Act, then print)	Machine (Print, then act)	Machine (Act without printing)
Overprint	+	x'01'	
Space one line	blank	x'09'	
Space two lines	0	x'11'	
Space three lines	-	x'19'	
Skip to channel 1	1		x'8B'

Any carriage control values other than those specified in this table are treated, for the purposed of determining record offset values, the same as the “Space one line” action.

Note: When the 390 indexer encounters a record in the load file containing a *x'89'* carriage control character (skip to channel 1 for “print, then act” support), the indexer changes the carriage control character to a *x'09'*. It then inserts a new record containing only a *x'8B'* carriage control character.

For purposes of determining the record offset values for both **LINEOFFSET=**ASREAD and **LINEOFFSET=**ASPRINTED, the *x'8B'* carriage control character is treated as a counted line. For example, if the load file contained these three lines:

```
(contains x'8B' in column 1)
 ACCOUNT 777777 (contains x'19' in column 1)
 JOHN SMITH (contains x'11' in column 1)
```

Using **ASREAD**, the indexing parameters to collect the account number and name would be as follows:

```
TRIGGER1=*,1,X'8B',(TYPE=GROUP)
FIELD1=1,10,6,(TRIGGER=1,BASE=0)
FIELD2=2,2,10,(TRIGGER=1,BASE=0)
```

Using **ASPRINTED**, the indexing parameters to collect the account number and name would be as follows:

```
TRIGGER1=*,1,X'8B',(TYPE=GROUP)
FIELD1=1,10,6,(TRIGGER=1,BASE=0)
FIELD2=4,2,10,(TRIGGER=1,BASE=0)
```

MCC2ANSI

All line print reports captured by the 390 indexer must include carriage control characters in column 1 of each line. These carriage control characters can be either Machine Code (MCC) or ISO/ANSI (ANSI). The **MCC2ANSI** parameter enables converting MCC carriage control to ANSI at capture time.

Parameters

When MCC2ANSI=Y is specified

If MCC carriage control characters are detected, they are converted to ANSI carriage control characters. This is done before performing any segmentation and indexing actions. Records that use the MCC NOP carriage control character (x'03') are ignored and not captured when MCC2ANSI=Y is specified.

When MCC2ANSI=N is specified

No conversion of carriage control characters is performed.

The default value is MCC2ANSI=N. If a value other than Y or N is specified for this parameter, an error message is issued and the load terminates.

Triggers

Identifies locations and string values required to uniquely identify the beginning of a group and the locations and string values of fields that are used to define indexes.

You must define at least one trigger and can define up to 144 triggers.

Syntax

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

Options and values

n
The trigger parameter that is identified. When you add a trigger parameter, use the next available number beginning with 1.

record
The input record where the 390 indexer locates the trigger string value. A record value of * (asterisk) indicates that the 390 indexer searches every input record for the trigger string value. For TRIGGER1 the input record must be * (asterisk). When a number is specified, the number is relative to the first line of the current page. The supported range of record numbers is 0 (the first line of the page) to 255.

column
The beginning column where the 390 indexer locates the trigger string value. The supported range of column numbers is 1 to 32756. If you specify an * (asterisk) or 0, the 390 indexer scans the record from left to right and searches for the trigger string value. A value of 1 refers to the first byte of the record, where the carriage control characters reside.

value
The string value that the 390 indexer uses to match the input data. It is recommended that you enter this value as a hexadecimal string.

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. Triggers are values that must match the values on a page for that page to be identified as the first page of a new document. You can also use the INDEXn parameter with BREAK=YES to identify the start of a new document. This indicates values on a page that must change to identify that page as the start of a new document.

FLOAT
Triggers that are used to locate index values that do not necessarily occur in the same location on each page. FLOAT triggers can use a value of * (asterisk) or a number for either the record or column values.

The **TYPE=GROUP,RECORDRANGE=**(*start*,*end* format is not supported by the 390 indexer.

XEROX DJDE Support

When the report being captured contains DJDE records, you must specify the DJDETRIG, DJDECOL, and DJDECNT indexing parameters . All records identified as "global DJDE records" are stored as document resources. Refer to each parameter for further details.

You must select a Data Type of **Global DJDE** from the **View Information** tab of the Application Definition window in the administration client when defining a DJDE application to Content Manager OnDemand.

Using the 390 indexer

Content Manager OnDemand application

You must specify to the ARSLOAD program that the 390 indexer process is to be used to capture a report.

About this task

You specify the name of the indexer on the **Indexer Information** page in the Content Manager OnDemand application. The name of the 390 indexer in the Content Manager OnDemand application is 390.

Procedure

To specify the indexer:

1. Start the administrative client.
2. Log on to the server.
3. Add an application.
4. Click the **Indexer Information** tab.
5. On the **Indexer Information** page, specify 390 in the **Indexer** field.

Large objects and the 390 indexer

The 390 indexer provides Content Manager OnDemand large object support for line print and AFP reports. In general, large objects can be used for all line print and AFP reports but should be considered for most documents that exceed 100 pages in size.

Non-Large Object documents are limited in size by available processor storage. The entire document is stored in memory during the load process and during the retrieval process. Excessively large documents can result in high storage requirements and high paging and should be considered as candidates for Large Object.

The ARSLOAD program in a z/OS environment

When using the ARSLOAD program to invoke the 390 indexer on the z/OS platforms, consider the following information.

When using JCL to run the ARSLOAD program to capture reports:

- Specify the **-s ddname** parameter to indicate the DD statement that points to the input report file that is being captured.
- Specify the name of a temporary file as the last parameter. The ARSLOAD program uses the temporary file for work space during the load process.
- Specify the **-j ddname** parameter to provide additional indexing instructions via a DD Name in the JCL. These parameters will be concatenated to the end of the parameter list from the application definition. If duplicate parameters are found, the last one specified is used.

ARSLOAD stores the document being processed in the directory selected for temporary file use. Analyze the directory to make sure that there is adequate additional space to hold the largest document that may be captured.

The directory to be used by ARSLOAD for temporary files is determined in the order:

- -c option in the ARSLOAD parameters
- Environment variable ARS_TMP
- Environment variable TEMP
- Environment variable TMP
- Current working directory if none of the previously specified directories

The following is an example of a parameter list for ARSLOAD. The DD statement that points to the input report file that is being captured is `OBJINPT`, the name of the application group is `CHKOPL1 01`, and `tempname` is the name of the file that is used for temporary work space.

```
//    PARM=('/-u SYSADMIN -p SYSADMIN -h ARCHIVE -n -v -s OBJINPT
//             -g "CHKOPL1 01" tempname')
```

Chapter 4. 400 indexer

The Content Manager OnDemand 400 indexer is the only Content Manager OnDemand indexer used for IBM i spooled files. The indexer is supported by IBM i, and cannot be used on any other platform. The 400 indexer is called by the **ADDRPTOND** command for SCS, SCS-extended, Advanced Function Presentation (AFP), and Line spooled files. You use the OnDemand Administrator client's graphical indexing tool to define the index criteria that the 400 indexer uses to locate and create index data for your spooled files.

The graphical tool can be invoked in one of two ways:

- By clicking the Select Sample Data button within the Report Wizard, or
- Selecting Sample Data and clicking the Modify button on the Indexer Information panel while creating a Content Manager OnDemand application definition

Indexing parameters include information that allows Content Manager OnDemand to identify key items in the print data stream and create index elements pointing to these items. You can specify the index information that allows Content Manager OnDemand 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. Content Manager OnDemand 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.

Index data is made up of an attribute name (for example, Customer_Name) and an attribute value (for example, Frank Booth), with a defined tag that identifies the location of the data on the print page. For example, the Account_Number tag with the pointer 1,21,16 means Content Manager OnDemand can expect to find Account_Number values starting in column 21 of specific input records. Content Manager OnDemand collects 16 bytes of information starting at column 21 and adds it to a list of attribute values found in the input. Content Manager OnDemand creates an index file when you index report files. The index file includes index elements that contain the offset and length of a group. Content Manager OnDemand calculates an index element for every group found in the input file. Content Manager OnDemand then writes the attribute values extracted from the input file to the index file.

Content Manager OnDemand uses the 400 indexer for SCS, SCS-extended, AFP, and Line spooled files. See the Report Wizard section in the Introduction of the *IBM Content Manager OnDemand for i: Common Server Administration Guide* for more information on the Report Wizard. See the section on Adding the Application in the Examples chapter of the *IBM Content Manager OnDemand for i: Common Server Administration Guide* for more information on defining an application without using the Report Wizard.

400 indexer parameters

Indexing parameters can contain indexing, conversion, and resource collection parameters, options, and values. For most reports, Content Manager OnDemand requires three indexing parameters to extract or generate index data:

- **TRIGGER** Content Manager OnDemand uses triggers to determine where to locate data. A trigger instructs Content Manager OnDemand to look for certain information in a specific location in the report file. When Content Manager OnDemand finds a record in the data stream that contains the information specified in the trigger, it can begin to look for index information.
  - Content Manager OnDemand compares data in the report file with the set of characters specified in a trigger, byte for byte.
  - A maximum of 16 triggers can be specified.
- **FIELD** The field parameter identifies the location, offset, and length of the data Content Manager OnDemand uses to create index values.

  - Field definitions are based on TRIGGER1 by default, but can be based on any of 16 TRIGGER parameters.
  - A maximum of 128 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 Content Manager OnDemand generates. For the group-level indexes Content Manager OnDemand stores in the database, you should name the attributes the same as the application group database field names.
  - Content Manager OnDemand can create indexes for a page, group of pages, and the first and last sorted values on a page or group of pages. Content Manager 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, you can move to a specific page by using the page-level indexes.

    Important: Page-level indexes must already exist in the AFP spooled file before indexing.
  - You can concatenate field parameters to form an index.
  - A maximum of 128 index parameters can be specified.

Content Manager OnDemand creates a new group and extracts new index values when one or more of the index values change, or the GROUPMAXPAGES value is reached.

```
  ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9
01                                                                    Page 0001
1
2                                       Jack Straw
3                                       4 Buxanchange Way
4                                       Wichitaw KS 99999-9999
5
6                                       Statement Date: 06/15/07
7                                       Account Number: 1234-5678-9876-0000
8
9                                       Balance: $2,984.17
```

Figure 3. Indexing a report

The following indexing parameters can be used to generate index data for the report shown in Figure 3 on page 176. The TRIGGER definitions tell the 400 indexer how to identify the beginning of a group in the input. The 400 indexer requires one TRIGGER definition to identify the beginning of a group (statement) in the sample file. For example:

- TRIGGER1 looks for the string `Page  0001` in column 72 of every record.

The trigger uniquely identifies the start of a statement in the report.

The FIELD definitions determine the location of the 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 the statement date index values, beginning in column 56 of the sixth record following the TRIGGER1 record.
- FIELD3 identifies the 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.

The following table lists the maximum values for certain indexing attributes:

Indexing attribute	Maximum value
Maximum number of lines per spooled file page (greater than printer file maximum due to allowance for overprint lines)	512
Maximum page width (positions per line)	378
Maximum number of triggers per page (for documents not using multi-key)	16
Maximum number of index values per page (for documents not using multi-key)	128
Maximum number of fields per page (for documents not using multi-key)	128
Maximum number of triggers per page (for multi-key documents)	512
Maximum number of index values per page (for multi-key documents)	1024
Maximum number of fields per page (for multi-key documents)	1024
Maximum number of index values per group (document) (for multi-key documents)	9999
Maximum size of an AFP resource segment (practical limit determined by disk space and memory on your server and workstations)	2 TB
Maximum size of any single AFP resource (practical limit determined by disk space and memory on your server and workstations)	2 TB

Important: For the Maximum page width indexing attribute, when using the **PRTTXTOND** command with **STMF(*NONE)**, which directs the output to a spooled file instead of a stream file, the maximum page width is 372.

Unique indexing parameter reference

To provide the segmentation and indexing instructions, the 400 indexer uses standard ACIF parameters and additional parameters that are unique to the 400 indexer. The following parameters are available for use.

CC

The **CC** parameter is set to YES by default by the graphical indexer when the 400 indexer is specified. If present, the **CC** parameter should not be changed or removed.

Syntax

```
CC=YES
```

CCTYPE

The **CCTYPE** parameter is set to A by default by the graphical indexer when the 400 indexer is specified. If present, the **CCTYPE** parameter should not be changed or removed.

Syntax

```
CCTYPE=A
```

CONVERT

The **CONVERT** parameter is not used by the 400 indexer, but it might be present in the indexer parameters. If present, the **CONVERT** parameter should not be changed or removed.

Syntax

```
CONVERT=NO | YES
```

CPGID

Required
No

Default Value
The default set by the graphical indexer is the code page of the Content Manager OnDemand instance into which the Content Manager OnDemand application definition is being created. However, if you remove the **CPGID** parameter from the indexer parameters and then you move to the view page of the application definition in the Content Manager OnDemand Administrator client, the code page is set to 850, which is ASCII. This causes your load to fail, which is by design, since you do not want to load System i data with code page set to 850.

Data Type
AFP, Line, SCS, SCS-Extended

Identifies the code page of the index data. The CPGID must match the code page of the input data. For reasons stated in the Default Value section, the CPGID parameter should not be removed.

Syntax

```
CPGID=value
```

Options and values

The value can be:

850
The default IBM code page if the **CPGID** parameter is not specified.

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

DOCTYPE

Required
Yes

Default Value
Set by the graphical indexer to match the data type specified on the View Information page.

Data Type
AFP, Line, SCS, SCS-Extended

Identifies the data type.

Syntax

```
DOCTYPE=value
```

Options and values

The *value* can be:

- AFP

- LINE
- SCS
- SCSEXT

FIELD

Identifies the location of the index data and can provide default and constant index values.

Required
Yes

Default Value
(None)

Data Type
AFP, Line, SCS, SCS-Extended

You must define at least one field. You can define up to 128 fields. The 400 indexer supports the following types of fields:

- Trigger field, which is based on the location of a trigger string value.
- Constant field, which allows you to provide the actual index value that is stored in the database.
- Transaction field, which you can use to index input data that contains one or more columns of data sorted in ascending sequence (from lowest to highest). Because it is not always practical to store every index value in the database, the 400 indexer extracts the first and last sorted values in each group. The data is sorted according to the collating sequence of the code page.
- Mask field, which uses a mask to match data from the field columns.

Trigger FIELD syntax

```
FIELDn=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). The value must be between 1 and 128.

record
The relative record number from the trigger on which the field is based. This is the record number where the 400 indexer begins to search for the field. The supported values include any number from -255 to 255, including 0.

column
The relative column number from the BASE. This is the column number where the 400 indexer searches 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. If you specify BASE=0, the column value can be 1 to 378. If you specify BASE=TRIGGER, the column value can be -378 to 378.

length
The number of contiguous bytes (characters) that compose the field. The supported range of values is 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, the 400 indexer adds padding blanks to fill out the field. Depending on the code page of your data, some characters might require more than one byte of data for each printed character. For example, DBCS data requires at least two bytes for every character. Data stored in an instance enabled with UTF-8 (Unicode) support requires one, two, or three bytes for each character, depending on the code page of the original data. See the *IBM Content Manager OnDemand for i: Common Server Planning and Installation Guide* for more information on UTF-8 support.

TRIGGER=*n*
Identifies the trigger parameter the 400 indexer 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 the 400 indexer uses the starting column number of the trigger string value to locate the field data. Choose from 0 (zero) or TRIGGER. If BASE=0, the 400 indexer adds zero to the field column offset. If BASE=TRIGGER, the 400 indexer 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. In the example, the starting column number of the field is always 10 bytes from the starting column number of the trigger. Specify BASE=TRIGGER and a column offset of 10 so that the 400 indexer correctly locates the field, regardless of the starting column of the trigger string value.

DEFAULT=*X'value'*
Determines the default value for the index when the field is not found in the page group. The default value must be specified as an EBCDIC hexadecimal string. For example, *X'value'*.

Example
The following field parameter causes the 400 indexer to locate field values that begin in column 83 of the same record that contains the TRIGGER1 string value. The field length is 8 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 the 400 indexer to locate field values that begin 10 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 the 400 indexer to locate the field by adding 10 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
A constant field is a field for which you specify the actual index value that is stored in the database.

```
FIELDn=constant
```

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 the 400 indexer extracts from a document by using a trigger field. The 400 indexer allows a constant field to be concatenated in an index with a field that is based on either a group trigger or 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). The value must be between 1 and 128.

constant
The literal (constant) string value of the field. This is the index value that is stored in the database. Specify a hexadecimal value using the format *X'constant'*, where constant is hexadecimal data. The constant value can be 1 to 250 bytes in length. The 400 indexer does not check the actual content of the supplied data for validity.

Examples
The following field parameter causes the 400 indexer to store the same string of hexadecimal characters in each INDEX3 value it creates.

```
FIELD3=X'F0F0F0F0F0F0F0F0F0' /* 000000000 */
INDEX3=X'D5D6D6D7',FIELD3,(TYPE=GROUP,BREAK=NO) /* NOOP */
```

The following field parameters cause the 400 indexer to concatenate a constant value with the index value extracted from the data. The 400 indexer concatenates the constant value that is 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 5-byte prefix (0000-) to all account numbers extracted from the data.

```
FIELD3=X'F0F0F0F060' /* 0000- */
FIELD4=0,66,14
INDEX3=X'818383A36D95A494',FIELD3,FIELD4,(TYPE=GROUP,BREAK=YES) /* acct_num */
```

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 value must be between 1 and 128.

*****
The record number where the 400 indexer begins searching for the field. A transaction field must specify an asterisk, meaning the 400 indexer searches every record in the group.

*****
The column number where the 400 indexer begins searching for the field. A transaction field must specify an asterisk. The OFFSET specification determines the column or columns where the 400 indexer locates the field.

Note: When defining a transaction field, if you enter a value other than an asterisk, the 400 indexer ignores the value. When you specify the OFFSET keyword of the **FIELD** parameter, the 400 indexer always uses the starting column number from the OFFSET keyword to determine the location of the field value.

length
The number of contiguous bytes (characters) that compose the field. The supported range of values is 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, the 400 indexer adds padding blanks to complete the field.

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. Each pair represents one column of data on the report, so if your report only has one column of sorted data, you have only one pair of start and end values. If you have more than one pair, 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 the 400 indexer will not detect a match.

MASK='*@#=¬^%'
Determines the pattern of symbols that the 400 indexer matches with data located in the field columns. When you specify the MASK keyword, you must also specify the OFFSET keyword. Valid mask symbols include the following symbols:

Table 15. Mask symbols

Mask symbol	Meaning
*	Not literal; matches a user-defined mask. Refer to the USERMASK.
@	Matches alphabetic characters.
#	Matches numeric characters.
¬	Matches any non-blank characters.
^	Matches any non-blank character.
%	Matches the blank character and numeric characters.
=	Matches any character.

The symbols in the mask are assumed to be in the code page that is specified by the CPGID parameter. The 400 indexer translates all characters in the MASK value except the MASK symbols shown in the preceding table. The 400 indexer then matches the input characters against the MASK value. For example, the following definitions

```
CPGID=37
FIELD3=*,*,8,(OFFSET=(10:17),MASK='A####-##',ORDER=BYROW)
```

cause the 400 indexer to search columns 10 through seventeen for an uppercase A (a hexadecimal C1) followed by four numeric characters (hexadecimal F0-F9), a dash (hexadecimal 60), and two numeric characters (hexadecimal F0-F9).

ORDER={BYROW|BYCOL}

Identifies where the 400 indexer can locate the smallest value and the largest value of a group of sorted values (sorted in ascending order) arranged in either rows or columns on the page. The default ORDER is BYROW. For ORDER=BYROW, the 400 indexer 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, the 400 indexer 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
```

Example

The following field parameter causes the 400 indexer to locate a 10 character numeric string that begins in column three of any records in the group. This format of the **FIELD** parameter is used to create indexes for the beginning and ending sorted values of each group.

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

Related information

"Defining transaction fields" on page 202

Mask FIELD syntax

The MASK keyword of the **FIELD** parameter is supported for fields located by either group triggers or float triggers, and also for transactions fields.

Unlike some of the other indexers for other Content Manager OnDemand platforms, the 400 indexer allows an **INDEX** parameter that is based on the field to include other fields.

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=X'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 the 400 indexer begins to search for the field. The supported values include any number from `-255` to 255, including `0`.

column
The relative column number from the BASE (specified with the BASE keyword of the **FIELD** parameter). This is the column number where the 400 indexer begins to search for the field. A value of 1 (one) refers to the first byte in the record. For files that contain carriage-control characters, column one refers to the carriage-control. If you specify BASE=0, then the column value can be from 1 to 378. If you specify BASE=TRIGGER, then the column value can be from `-378` to 378. If the specified value exceeds the physical length of the record, then the 400 indexer reports an error condition and terminates processing.

length
The number of contiguous bytes (characters) that compose the field. The supported range of values is 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, the 400 indexer adds padding blanks to complete the field.

TRIGGER=*n*
Identifies the trigger parameter that the 400 indexer uses to locate the field.

BASE=*{0|TRIGGER}*
Determines whether the 400 indexer 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 the 400 indexer adds zero to the field column offset. If you specify BASE=TRIGGER, then the 400 indexer adds the starting column number of the trigger string value to the field column offset. You must specify BASE=0 if the field data always starts in a specific column. You must 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 10 bytes from the starting column number of the trigger; you would specify BASE=TRIGGER and a of 10 (ten) so that the 400 indexer correctly locates the field, regardless of the starting column of the trigger string value.

MASK='*@#=¬^%'
Specifies a pattern of symbols that the 400 indexer uses to match data in the field columns. If the data matches the MASK, then the 400 indexer selects the field. You can specify the following symbols in the MASK:

Table 16. Mask symbols

Mask symbol	Meaning
*	Not literal; matches a user-defined mask. Refer to the USERMASK.
@	Matches alphabetic characters.

Table 16. Mask symbols (continued)

Mask symbol	Meaning
#	Matches numeric characters.
¬	Matches any non-blank characters.
^	Matches any non-blank character.
%	Matches the blank character and numeric characters.
=	Matches any character.

For example, given the following definitions:

```
TRIGGER2=*,25,'SOURCE',(TYPE=FLOAT)
FIELD2=0,38,4,(TRIGGER=2,BASE=0,MASK='####',DEFAULT=X'F1F0F0F0')
```

The 400 indexer selects the field only if the data in the field columns contains numeric characters.

DEFAULT=*X'value'*

The default value is used if the field is not found. This occurs when the trigger locating the field is not found. The default value can be from 1 to 250 characters in length.

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

Related reference

Assigning default index values

You can create a Content Manager OnDemand application definition with an index field that does not always exist on the print page. If a value is not found for that field during indexing (if the field location does not exist on the particular print page), then the DEFAULT keyword is used to determine the default value to use. The DEFAULT keyword can be placed on the FIELD indexer parameter line of the indexer parameters for a particular application definition.

Regular expression FIELD syntax

To specify a trigger field, you must assign the field a number prefixed by **FIELD**, then specify the location of the field on the report.

Multiplatform syntax:

```
FIELDn=record,column,length,(TRIGGER=n,BASE={0 | TRIGGER},[REGEX='regular expression'][,DEFAULT=X'value]')
```

REGEX='regular expression'

The 400 indexer extracts the text specified by the column and length values. After the field is extracted, the 400 indexer applies the regular expression to the text. Any text that matches the regular expression is extracted for the field. If the matching text is shorter than the length specified in the **FIELD**, it is padded with blanks until it equals the length.

If the record is only long enough to contain part of the field, the regular expression is applied only to the portion of the record that is present.

The regular expression must be specified in the code page given by the **CPGID** parameter. It can be specified in hexadecimal.

The maximum length of the regular expression is 250 bytes. A mask and a regular expression cannot both be specified on the same **FIELD** parameter.

FILEFORMAT

The **FILEFORMAT** parameter is always RECORD when using the 400 indexer, and if present, must not be changed or removed.

Syntax

```
FILEFORMAT=RECORD
```

IMAGEOUT

The **IMAGEOUT** parameter is set to ASIS by default by the graphical indexer when the 400 indexer is specified. If present, the **IMAGEOUT** parameter should not be changed or removed.

Syntax

```
IMAGEOUT=ASIS
```

ASIS is the only value supported by the 400 indexer is for the **IMAGEOUT** parameter.

INDEX

Identifies the index name and the fields on which the index is based. You must define at least one index parameter.

Required
Yes

Default Value
(None)

Data Type
AFP, Line, SCS, SCS-Extended

Identifies the index name, the field or fields on which the index is based, and the type of index the 400 indexer generates. You can define group indexes for SCS, AFP, and line data. You must define at lease one index parameter. You can define up to 128 index parameters. When you define a group index, IBM recommends that you name the index the same as the application group database field name.

Syntax

```
INDEXn=name,FIELDnn[,...FIELDnn],TYPE=type)][,(ALLOWMULTIPLEVALUES={NO|YES})]
```

Options and values

n
The index parameter identifier. When you add an index parameter, use the next available number beginning with 1.

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, 00123456789. The index name can be a maximum of 250 bytes in length. 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'*. The creation of the hexadecimal data is done for you automatically by the graphical indexer.

FIELDnn
The name of the field parameter or parameters the 400 indexer uses to locate the index. A maximum of 128 field parameters can be specified. Separate field parameter names with a comma. The total length of the contents of all the specified field parameters cannot exceed 250 bytes. For example, the

value of FIELD1 might be "(888) " and the value of FIELD2 might be "555-1212". If INDEX1 is defined as

```
INDEX1 = X'D788969585',FIELD1,FIELD2,(TYPE=GROUP)
```

then the values of FIELD1 and FIELD2 is concatenated into INDEX1. The total length of the concatenated string cannot exceed 250 bytes. GROUPRANGE indexes must name only one transaction field. GROUPRANGE indexes cannot break a group - you must specify BREAK=NO.

TYPE=*type*

The type of index the 400 indexer generates. The default index type is GROUP. The following list has valid index types:

- `TYPE=GROUP[,BREAK={YES|NO}][,(ALLOWMULTIPLEVALUES= {NO|YES})]`

  Create a group index value. The 400 indexer creates one index value for each group. You can specify whether the 400 indexer includes or ignores the index when calculating a group break. When BREAK=YES (the default), the 400 indexer begins a new group when the index value changes. For most reports, BREAK must always be set to YES. BREAK=NO is useful when you define two or more indexes and you want the 400 indexer to begin a new group only when a specific index value changes. Specify BREAK=YES for the index that you want the 400 indexer to use to control the group break. Specify BREAK=NO for the other indexes. A GROUP index that specifies BREAK=NO can also specify ALLOWMULTIPLEVALUES=YES if you want the 400 indexer to collect multiple values for this index for each group (document). This concept is also known as multi-key. The default is NO, which causes the 400 indexer to collect only the first occurrence of an index within a given group. Specify YES if you want multiple values for this index to be collected per group. An example of the use of this multi-key capability might be for an invoice that has multiple part numbers that are listed on the detail lines of the invoice. Without the multi-key capability, the 400 indexer would only collect the first one of the part numbers on the page. Multi-key tells the 400 indexer to continue to collect multiple part numbers before moving on to the next invoice. See "Defining multi-key indexes" on page 199 for more information.

- `TYPE=GROUPRANGE,BREAK=NO`

  Create group indexes. The 400 indexer creates index values for the first and last sorted values in each group. The 400 indexer 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. The 400 indexer assumes that the input values are sorted. You can define one GROUPRANGE index per report. A GROUPRANGE index must name 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, the 400 indexer uses the value of the **GROUPMAXPAGES** parameter to determine the number of pages in a group. For example, you need to index a 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, the 400 indexer will use the value of the GROUPMAXPAGES parameter to determine when to close the current group and begin a new group.

Examples

Group index

The following index parameter causes the 400 indexer to generate group indexes for date index values. The index type is optional, but defaults to group. When the index value changes, the 400 indexer closes the current group and begins a new group.

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

The following index parameters cause the 400 indexer to generate group indexes for customer name and account number index values. The index type is optional, but defaults to group. The 400 indexer 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 the 400 indexer to generate grouprange indexes for loan number index values. The 400 indexer extracts the beginning and ending loan numbers in each group of pages. 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 is used to determine the number of pages in each group. The 400 indexer 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
```

INDEXOBJ

Determines the level of indexes the 400 indexer includes in the index file.

Syntax

Required

No

Default Value

GROUP

Data Type

AFP, Line, SCS, SCS-Extended

```
INDEXOBJ=value
```

Options and values

The value can be:

GROUP

The 400 indexer includes group-level index entries in the index file.

ALL

The 400 indexer includes group-level and page-level indexes in the index file. You must specify INDEXOBJ=ALL for reports that require Large Object support.

INDEXSTARTBY

Determines the page number by which the graphical indexer must find a group indexing field.

Syntax

Required

Yes

Default Value

1

Data Type

AFP, Line, SCS, SCS-Extended

A group indexing field is a field that is based on a group or recordrange trigger. The graphical indexer issues a warning message if it does not find all group triggers before the specified page number. The **INDEXSTARTBY** parameter is set to 1 by default by the graphical indexer when the 400 indexer is specified, but can be changed if necessary by clicking the indexer properties icon while defining triggers, fields, and indexes in the graphical indexer.

The **INDEXSTARTBY** parameter is not used by the 400 indexer when locating triggers and fields.

```
INDEXSTARTBY=value
```

Options and values
The *value* is a number from 1 to 99.

INDEXSTYLE

For definitions migrated from a Content Manager OnDemand Spool File Archive environment, INDEXSTYLE indicates whether the original Spool File Archive definition was a DOC, PAGE, or NODX report type. If not specified, DOC is assumed.

Syntax

Required
No

Default Value
(None)

Data Type
AFP, Line, SCS, SCS-Extended

```
INDEXSTYLE=value
```

Options and values
The *value* can be:

- DOC
- PAGE
- NODX

STARTINDEXINGONPAGE

Indicates the page number on which the OS/400® indexer begins indexing. Any pages before the starting page are discarded.

Syntax

Required
No

Default Value
1

Data Type
Line, SCS, SCS-Extended

```
STARTINDEXINGONPAGE=value
```

Options and values
The *value* is a number from 1 to 99.

STARTTRANSACTIONFIELDSONLINE

Indicates the line number on which the OS/400 indexer begins searching for transaction fields.

Syntax

Required
No

Default Value
1

Data Type
AFP, Line, SCS, SCS-Extended

```
STARTTRANSACTIONFIELDSONLINE=value
```

Options and values
The *value* is a number from 1 to 510.

STARTTRIGGERSONLINE

Indicates the line number on which the OS/400 indexer begins searching for triggers.

Syntax

Required
No

Default Value
1

Data Type
AFP, Line, SCS, SCS-Extended

```
STARTTRIGGERSONLINE=value
```

Options and values
The *value* is a number from 1 to 510.

TRANSLATEPRINTCONTROL

Indicates whether the 400 indexer should process carriage control characters before determining the line number on which to locate triggers, fields, and indexes.

Syntax

Required
No

Default Value
NO

Data Type
Line, SCS, SCS-Extended

The **TRANSLATEPRINTCONTROL** parameter must be added manually to your indexer parameters by clicking the **keyboard radio** button and then **modify** button in the Parameters Source are of the Indexer Information page of your Content Manager OnDemand Application definition.

```
TRANSLATEPRINTCONTROL=value
```

Options and values
The *value* can be:

- YES
- NO

Related information
"Understanding Translate Print Control" on page 205

TRIGGER

Identifies locations and string values required to uniquely identify the beginning of a group and the locations and string values of fields used to define indexes. You must define at least one trigger and can define up to 16 triggers.

Required
Yes

Default Value
(None)

Data Type
AFP, Line, SCS, SCS-Extended

Syntax

TRIGGERn=*record*,*column*,*value*|**REGEX='** *regular expression*'[,(**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 the 400 indexer locates the trigger string value. For TRIGGER1 and float triggers, the input record must be * (asterisk), so that the 400 indexer 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
If the 400 indexer is using a value, then this is the beginning column where the 400 indexer locates the trigger string value.

The supported range of column numbers is 0 to 378. To force the 400 indexer to scan every record from left to right for the trigger string value, specify an *(asterisk) 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 following examples.

If the 400 indexer is using a regular expression, then this is the beginning column where the 400 indexer looks for text that matches the regular expression. The regular expression must match text which begins in the specified column. If a column range is specified, then the 400 indexer will only search the columns in the column range for the text that matches the regular expression. The regular expression must match text which begins in one of the columns specified by the column range.

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 the 400 indexer uses to match the input data. The string value is case sensitive. Enter the value in hexadecimal. The value can be from 1 to 250 bytes in length. You can specify either a value or a regular expression, but not both.

REGEX='*regular expression*'
The regular expression that the 400 indexer uses to match the input data. The regular expression must be specified in the code page given by the CPGID parameter, and can be from 1 to 250 bytes in length. The regular expression can be specified in hexadecimal. You can specify either a value or a regular expression, but not both.

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. The 400 indexer determines the location of the field by searching the specified range of records. The range can be from 0 to 255. The 400 indexer 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, the 400 indexer 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. The 400 indexer 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 the 400 indexer, 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 the 400 indexer 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. The 400 indexer 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 Content Manager OnDemand, each indexed group of pages is known as a *document*. When you index an input file and load the data into the system, Content Manager OnDemand stores the group indexes that are generated by the 400 indexer into the database and stores the documents on storage volumes. Content Manager OnDemand uses the group indexes to determine the documents that match the search criteria that is entered by the user.

Notes

1. At least one TRIGGERn or FIELDn value must exist on the first page of every unique page group. The 400 indexer cannot detect an error condition if TRIGGERn or FIELDn is missing, but the output might be incorrectly indexed.
2. TRIGGER1 must be specified when the 400 indexer is requested to index the file.

Related concepts

ACIF indexer parameters

Depending on whether you run ACIF on a multiplatform (Linux, UNIX, or Windows) or z/OS system, the defaults for certain parameters change. The defaults for the multiplatform and z/OS systems are provided in the parameter reference.

Examples

TRIGGER1

The following TRIGGER1 parameter causes the 400 indexer to search column one of every input record for the occurrence of a start of new page carriage control character. The record value for TRIGGER1 must be an asterisk. 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 the 400 indexer 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 trigger type is optional, but defaults to group. TRIGGER1 must be a group trigger.

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

Group trigger

The following trigger parameter causes the 400 indexer 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 trigger type is optional, but defaults to group.

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

Group trigger with column range

The following trigger parameter causes the 400 indexer 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 trigger type is optional, but defaults to group.

```
    TRIGGER2=6,50:52,X’C1838396A495A340D5A494828599’,(TYPE=GROUP)
```

Recordrange trigger

The following trigger parameter causes the 400 indexer to attempt to locate the string value `Account Number` beginning in column fifty within a range of records in each group. The TRIGGER3 string value can occur in records six, seven, or eight following TRIGGER1 in each group.

An asterisk must be used for record number. The 400 indexer uses the recordrange to determine which records to search for the trigger string value. The trigger type is optional, but must be GROUP for a recordrange trigger.

```
TRIGGER3=*,50,X'C1838396A495A340D5A494828599',(TYPE=GROUP,RECORDRANGE=(6,8))
```

Float trigger

The following trigger parameter causes the 400 indexer 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 trigger type is float and must be specified.

```
TRIGGER3=*,5,X'E3A8978540968640C99583969485',(TYPE=FLOAT)
```

Triggers using a regular expression

The following TRIGGER parameter examples use regular expressions to search for strings.

The following trigger parameter causes the 400 indexer to search for the string "PAGE 1". The search will start in column 1 and continue until the end of each record.

```
CPGID=37
TRIGGER1=*,*,REGEX='PAGE 1',(TYPE=GROUP)
```

The following trigger parameter causes the 400 indexer to search for a string containing four uppercase letters followed by three digits. The regular expression must match the text starting in column 10.

```
CPGID=1141
TRIGGER2=*,10,REGEX='[A-Z]{4}[0-9]{3}',(TYPE=FLOAT)
```

The following trigger parameter causes the 400 indexer to search in columns 15 through 18 for a string containing the letter "P" followed by three lowercase letters. The TYPE is GROUP by default.

```
CPGID=500
TRIGGER1=*,15:18,REGEX=X'D74A8160A95AC0F3D0'
/* regular expression is P[a-z]{3} */
```

BREAK setting

A group is a set of pages that logically belong together. For example, all the pages in a single bank statement could comprise a group. A group is a single document, or a *segment*, as it was known in Spool File Archive. A group break is the process of closing the current group and starting a new group. In Spool File Archive, this process was known as segmentation. For a specific group index, the BREAK setting determines whether the OS/400 indexer begins a new document when that index's value changes.

When you specify BREAK=YES, the OS/400 indexer begins a new group when the value of the field on which the index is based changes. BREAK=NO is useful when you define two or more fields and you want the OS/400 indexer to begin a new group only when the other of the two fields' value changes. Specify BREAK=YES only for the index that is based on the field that you want the OS/400 indexer to use to control the group break. Specify BREAK=NO for all the other indexes in the group.

To expand on the bank statement example, consider storing bank statements. Each statement begins with a change in account number from the previous statement. You defined indexes for Account Number, Customer Name, and Statement Date. Most likely, you want Account Number to be set to BREAK=YES, Customer Name to BREAK=NO, and Statement Date to BREAK=NO. Doing this ensures that a group break occurs only when Account Number changes. The corresponding indexer parameters in the Application definition might look like this:

```
INDEX1=X'C1838396A495A3D5A494828599',FIELD1,(TYPE=GROUP,BREAK=YES) /* AccountNumber
*/
INDEX2=X'C3A4A2A396948599D5819485',FIELD2,(TYPE=GROUP,BREAK=NO)    /* CustomerName
*/
INDEX3=X'E2A381A385948595A3C481A385',FIELD3,(TYPE=GROUP,BREAK=NO)  /* StatementDate
*/
```

The Content Manager OnDemand Administrator client's Report Wizard is designed to simplify the process of defining application groups, applications, and folders. The Wizard makes the assumption that any change in an index that is defined as TYPE=GROUP should cause a group break. Thus, it sets all index fields to BREAK=YES. If the index is based on a float trigger (TYPE=FLOAT), then BREAK=NO is set by default. Indexes based on float triggers can be changed to BREAK=YES, if necessary.

Note that if you have selected the Allow Multiple Values option, BREAK is automatically set to NO and should not be changed.

If you already archived data with all of your indexes set to BREAK=YES, you can still make this change. Changing some of your indexes from BREAK=YES to BREAK=NO can be done at any time. As with any change to your indexer parameters, you should verify that your reports archive correctly after the change.

Any reports already archived do not need to be rearchived; however, the change will only affect reports that are archived after the change is made.

Controlling maximum number of pages per group

You might want to set a maximum number of pages for each group that is indexed. Content Manager OnDemand can use the value of the GROUPMAXPAGES indexer parameter to determine the number of pages in a group. For example, you need to index a report consisting of thousands of pages of detail. If your BREAK=YES criteria do not result in small enough groups of pages (or segments) of the report, you can use GROUPMAXPAGES=100, for example, to force Content Manager OnDemand to close the current group and begin a new group for any group that reaches 100 pages. In other words, if the GROUPMAXPAGES value is reached before the value of a group index changes, Content Manager OnDemand forces the creation of a new group. If you do not specify a value for the GROUPMAXPAGES parameter, Content Manager OnDemand 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.

Using Group triggers versus Float triggers

When trying to decide whether to use a Group trigger or a Float trigger, consider the following information.

Use Group triggers when:

The trigger is always found on the same page as trigger 1.

The trigger is always found in the same row or range of rows relative to trigger 1.

The trigger is found more than once, but you only want to use the first occurrence.

Use Float triggers when:

The trigger might be found on the same page or on a different page as trigger 1.

The trigger is found more than once and you want to use all occurrences.

The index will be defined as multi-key.

Using a mask when defining application fields

A mask specifies the pattern of symbols that the indexing program matches with data located for a particular field.

With the 400 indexer, a mask can be used with either a trigger-based field or a transaction field. If the data matches the mask, then the indexer selects the field. If the data does not match the mask, then the field is treated as if the trigger or transaction field was not found.

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, a mask of ####.## would cause the indexer to select the field only if the data in the field (from left to right) contains four numeric characters, followed by a decimal point, followed by two numeric characters.

An example of the indexer parameter syntax for a field with a mask is as follows:

```
FIELD4=0,-24,7,(TRIGGER=3),BASE=TRIGGER,MASK='####.##')
```

Using regular expressions

A regular expression is a pattern that is used to match characters in a string. There are many online resources that explain the syntax rules of regular expressions.

Regular expression examples

The following examples show some common regular expressions:

Table 17. Common regular expressions

Regular expression	Results
Account	Finds the characters "Account." By default searches are case sensitive.
[A-Z]	Finds one uppercase letter.
[A-Z]{3}	Finds three consecutive uppercase letters.
[0-9]{5}	Finds five consecutive digits.
[0-9]+	Finds one or more digits.
[^a-z]	Finds everything except lowercase a to z.
\s	Finds one whitespace character (space, tab, and so on).
\S	Finds any character except for whitespace.

The 400 indexer can use a regular expression in the TRIGGER and FIELD parameter. In the TRIGGER, the regular expression specifies the pattern for which to search; in the FIELD, the regular expression is applied to the characters which have been extracted from the field in a way similar to using a mask.

The regular expression must be specified in the code page given by the CPGID parameter. The regular expression can be specified as text, for example:

```
CPGID=37
TRIGGER1=*,*,'PAGE',(TYPE=GROUP)
TRIGGER2=*,25,REGEX='[A-Z]{3}-[A-Z]{6}',(TYPE=FLOAT)
FIELD1=0,9,2,(TRIGGER=1,BASE=TRIGGER)
FIELD2=0,38,10,(TRIGGER=2,BASE=0,REGEX='[A-Z] [0-9]{3}-\S+')
INDEX1='Page',FIELD1,(TYPE=GROUP,BREAK=YES)
INDEX2='Sub-Source',FIELD2
```

In this example TRIGGER2 uses a regular expression, which specifies a pattern of three uppercase letters, followed by a hyphen, followed by six uppercase letters. The text "SUB-SOURCE" would match the pattern.

FIELD2 uses a regular expression, which specifies one uppercase letter, followed by a space, followed by three numbers, followed by a hyphen, followed by one or more non white space characters. The characters "Q 010-1", "I 000-RS", or "L 133-1B" would match this regular expression.

The regular expression can also be specified in hexadecimal in the code page given by the CPGID parameter, for example:

```
CPGID=500
TRIGGER1=*,1,REGEX=X'4AF060F95AC0F3D0'  /* [0-9]{3} */
```

Performance

All text to which the regular expression is applied is converted to UTF-16.

- Performance might not be as fast when you use a regular expression. Using a text string can be faster.

Regular expressions and the TRIGGER parameter

On the TRIGGER parameter use the regular expression instead of a text string. A regular expression can be used on both a group trigger and a floating trigger.

The maximum length of the regular expression is 250 bytes.

If an asterisk is specified for the column, the 400 indexer searches the entire record for the string that matches the regular expression. If a column is specified, the 400 indexer searches the text starting in that column for the string that matches the regular expression. The regular expression must match text which begins in that column. If a column range is specified, the 400 indexer searches only the text within the column range for the string that matches the regular expression. The regular expression must match text which begins in one of the columns specified by the column range.

When the regular expression matches the text in a record, the 400 indexer looks for the next trigger, or, if all the group triggers have been found, the 400 indexer collects the fields.

Regular expressions and the FIELD parameter

On the FIELD parameter use the regular expression instead of a mask. A mask and a regular expression cannot both be specified on the same FIELD parameter.

The maximum length of the regular expression is 250 bytes.

The regular expression can be specified on a field based on a group trigger, a field based on a floating trigger, or a transaction field.

The 400 indexer extracts the text specified by the column and length values. The maximum length of a field that can be specified in the FIELD parameter is 250 bytes. After the field is extracted, the 400 indexer applies the regular expression to the text. Any text that matches the regular expression is extracted for the field. If the matching text is shorter than the length specified in the FIELD parameter, it is padded with blanks until it equals the length.

Default values for fields

If the regular expression does not match any text in the field, a default value can be used. Whether a default value is used and which type of default value depends on the type of field. There are three types of fields: fields based on group triggers, fields based on floating triggers, and transaction fields.

Group field

1. If a regular expression does not match any text in the group field, the default value specified on the FIELD parameter is used.
2. If the record is only long enough to contain part of the field, the regular expression is applied only to the portion of the record that is present.

Floating field

1. If a regular expression does not match any text in the floating field, there is no error and the default value specified on the FIELD parameter is not used.
2. If the record is only long enough to contain part of the field, the regular expression is applied only to the portion of the record that is present.
3. In the case of (1) the load process can use the default value in the Application, Load Information.

Transaction field (grouprange or pagerange field)

1. If the regular expression does not match any text in the transaction field, there is no error and processing continues. A default value cannot be specified for a transaction field.
2. If the record is not long enough to contain the entire field, no field is collected. There are no errors and processing continues.

Examples

Using a regular expression for a trigger:

```
TRIGGER1=*,1,REGEX='P[A-Z]{3} ',(TYPE=GROUP)
```

This regular expression will match text that begins in column 1 with the letter 'P' and is followed by three uppercase letters followed by a space. For example:

```
"PAGE "
```

Using a regular expression to extract a date in the form of "July 4, 1956":

```
TRIGGER1=*,1,'1'
FIELD1=0,13,18,(REGEX='[A-Z][a-z]+ [0-9]+, [0-9]{4}',DEFAULT='January 1, 1970')
INDEX1='Date',FIELD1
```

Using a regular expression with a transaction field to extract a range of Social Security numbers:

```
TRIGGER1=*,1,'1'
FIELD1=0,30,3
FIELD2=*,*,12,(OFFSET=(59:70),ORDER=BYROW,REGEX='[0-9]{3}-[0-9]{2}-[0-9]{4}')
INDEX1='DEPT',FIELD1,(TYPE=GROUP)
INDEX2='SOCIAL SECURITY NUMBER',FIELD2,(TYPE=GROUPRANGE)
```

Assigning default index values

You can create a Content Manager OnDemand application definition with an index field that does not always exist on the print page. If a value is not found for that field during indexing (if the field location does not exist on the particular print page), then the DEFAULT keyword is used to determine the default value to use. The DEFAULT keyword can be placed on the FIELD indexer parameter line of the indexer parameters for a particular application definition.

If the field location does not exist on the particular print page, and a default value is not specified, blanks will used. For string fields, this will allow the load to complete successfully. For date, decimal, and integer fields, this will result in the load failing.

The DEFAULT keyword can be specified in one of two ways. The first method allows you to specify an actual value (given in alphanumeric or hex format). The second method allows you to use the default value that you have specified on the Load Information tab of the Content Manager OnDemand application definition or to use index propagation (described in the following information).

Examples of the first method:

```
DEFAULT='your_Value'
```

(such as DEFAULT='ABC') or

```
DEFAULT=x'your_Hex'
```

(such as DEFAULT=x'C1C2C3')

Examples of the second method:

```
DEFAULT='_*USELOADDEFAULTORPROPAGATION*_'
```

or

```
DEFAULT=x'6D5CE4E2C5D3D6C1C4C4C5C6C1E4D3E3D6D9D7D9D6D7C1C7C1E3C9D6D55C6D'
```

(In this second case, the hex value specified is the hexadecimal representation of the character string _*USELOADDEFAULTORPROPAGATION*_.)

The second method (using _*USELOADDEFAULTORPROPAGATION*_ or its hexadecimal representation) allows the load process to assign the default value from the Load Information tab of the application definition or for propagation to occur. To have the load process assign a default from the Load Information tab, you must specify one by using the Content Manager OnDemand Administrator Client. If you have not specified a default, propagation occurs. Propagation is the process of carrying a value over from its previously found value. This can be useful but can also have unintended results. For example, if the field was a customer number, the value for customer number is carried from the previous document if one was not found for the current document. This might not be what you intend to happen. Exercise caution when using this second method, as propagation can occur.

Important: With propagation, the loading of the spooled file will fail if an actual field value is not found on the print page for the first document in the spooled file, since there would be nothing to propagate into that field.

Additionally, _*USELOADDEFAULTORPROPAGATION*_ cannot be used on a FIELD defined with a mask, such as MASK=##/##/##.

To further understand the use of _*USELOADDEFAULTORPROPAGATION*_, consider the following sample report. Notice that a value for the Code field does not appear on every printed page.

```
10/04/2006      Code:           K
                Account Number: 123456798
                Invoice Number: 876544-11

10/04/2006
                Account Number: 123456987
                Invoice Number: 876545-08
```

The indexer parameters might look something like this:

```
TRIGGER1=*,42,X'61',(TYPE=GROUP) /* / */
TRIGGER2=*,60,X'C39684857A',(TYPE=FLOAT) /* Code: */
FIELD1=0,40,10,(TRIGGER=1,BASE=0)
FIELD2=1,66,9,(TRIGGER=1,BASE=0)
FIELD3=2,66,9,(TRIGGER=1,BASE=0)
FIELD4=0,66,1,(TRIGGER=2,BASE=0,
  DEFAULT=X'6D5CE4E2C5D3D6C1C4C4C5C6C1E4D3E3D6D9D7D9D6D7C1C7C1E3C9D6D55C6D')
  /* _*USELOADDEFAULTORPROPAGATION*_ */
INDEX1=X'A2A381A385948595A36D8481A385',FIELD1,(TYPE=GROUP,BREAK=YES) /* statement_date */
INDEX2=X'81838396A495A36D95A494828599',FIELD2,(TYPE=GROUP,BREAK=YES) /* account_number */
INDEX3=X'8995A5968983856D95A494828599',FIELD3,(TYPE=GROUP,BREAK=YES) /* invoice_number */
INDEX4=X'83968485',FIELD4,(TYPE=GROUP,BREAK=NO) /* code */
```

If you need the value of 'K' found for the first account number/invoice number to be propagated to the second account number/invoice number, you would NOT specify a default for the Code field on the Load Information page of the Content Manager OnDemand application definition. The use of _*USELOADDEFAULTORPROPAGATION*_ without a default specified on the Load Information page of the application causes propagation to occur. If, instead, you need to specify a different default value (such as 'D') to be used when the field value is not found on the page, your indexer parameters would look exactly the same as shown in the preceding example, but you would specify the 'D' for Default Value on the Load Information page to cause Content Manager OnDemand to assign a 'D' to the Code field for the second account number/invoice number.

Handling SCS spooled files that have AFP overlays

The preferred method of handling SCS spooled files that have an AFP overlay named in their associated printer file is to simply change the DEVTYPE parameter of the printer file used to create the original spooled file to *AFPDS. This will cause IBM i to put the data into spool as *AFPDS, which is the most

efficient way for Content Manager OnDemand to capture (load) this type of spooled data. However, making this change will require the original, production spooled file to be printed on an AFPDS printer. In most cases, if you really are printing it with an overlay, then this should not be a problem. However, if you are printing it on a line printer with preprinted forms, this approach will not work.

If, for some reason you cannot change the original printer file's DEVTYPE parameter to *AFPDS, Content Manager OnDemand can do the conversion to AFP automatically, allowing the spooled file to be viewed and printed with fidelity. (This method is more time-consuming than letting IBM i do it using the DEVTYPE parameter of the printer file.) To enable this conversion, simply specify both the data type and the DOCTYPE indexer parameter in the Content Manager OnDemand Application definition as AFP rather than SCS. When Content Manager OnDemand encounters an *SCS spooled file that has an overlay, and the Application definition and DOCTYPE indexer parameter both specify AFP as the data type, Content Manager OnDemand will convert the *SCS data to *AFPDS and store that newly created *AFPDS spooled file. Reprints out of Content Manager OnDemand will require an AFP-capable printer, but that should be expected due to the overlay. If you specify a data type of AFP in your Content Manager OnDemand Application definitions for any other type of non-AFP spooled file, the loading of the data will fail.

Using Tag Logical Elements (TLEs)

Using Tag Logical Elements (TLEs) to identify index data requires no special check boxes or other special setup. The Content Manager OnDemand graphical indexer (which is invoked by the Content Manager OnDemand Administrator Client when defining an application) automatically displays TLE data at the top of each print page before displaying the data itself, allowing you to use the TLE data just as you use the print data itself to extract index information (such as a customer number or invoice number).

An example of the data you might see in the Content Manager OnDemand Administrator Client's graphical indexer when you are working with TLEs in an AFPDS spooled file is shown in the following example. The four lines near the top, immediately following the *GROUP_START line, represent the TLE information. The AFP datastream *text* must be encoded in EBCDIC and not ASCII. This is also true of TLEs.

```
*GROUP_START                    113928
Invoice Number                  113928
Invoice Date                    06/15/07
Customer Number                 44332
Invoice Total                   $   5,282.37

       ABC COMPANY
       101 Plagioclase Blvd.
       Deva Station        VA 55564

       528 555-1234

SHIP DATE    06/30/07
       XYZ Inc.
       P.O. Box 47899
       Ridiculous          TN 79832

       CUSTOMER NUMBER 44332

       PURCHASE ORDER NO. - C3050279

17 IGUANAS                 3.23       0.11        77.34
93 SHOE HORNS             18.95      13.13       127.83
55 RUNCIBLE SPOONS        43.43       9.23       239.01
55 HATRACKS               97.00      43.83     4,721.64
93 THELMIN WIRES           0.54       2.32        14.12
09 TOOTHPICKS             53.00      19.91       102.43

                                                5,282.37
```

Defining multi-key indexes

Multi-key indexes can be used when an index value occurs multiple times within a single document. For example, invoices might have invoice number, customer number, and customer name defined as the first three index fields, each occurring once within a given invoice. Then you might also want to define item

number as a multi-key index, since there might be multiple item numbers within one invoice. With multi-key support, an end-user could search by item number to find any invoice for a given item number, no matter where that item number appeared in the list of invoiced items. Without the multi-key capability, only the first item number on the page would be indexed.

To enable multi-key indexing, the keyword ALLOWMULTIPLEVALUES=YES must be added to each INDEX statement that is to have multiple values captured per document. For example:

```
INDEX2=X'97969596',FIELD2,(TYPE=GROUP,BREAK=NO,ALLOWMULTIPLEVALUES=YES)
```

The keyword would be added to the Content Manager OnDemand Application definition. Go to the Indexer Information tab, then click on Keyboard and then Modify to edit the Application's Indexer Parameters. Keyword ALLOWMULTIPLEVALUES is only valid when BREAK=NO. Also note that unlike the Content Manager OnDemand Spool File Archive multi-key rule, defining an index as multi-key does not require all subsequent index fields to also be defined as multi-key. In a Common Server environment, as is shown in the example, you can define an index as multi-key and then define another one below it that is not multi-key. However, a field used for a multi-key index must be found on or below the row containing the float trigger used to locate that field.

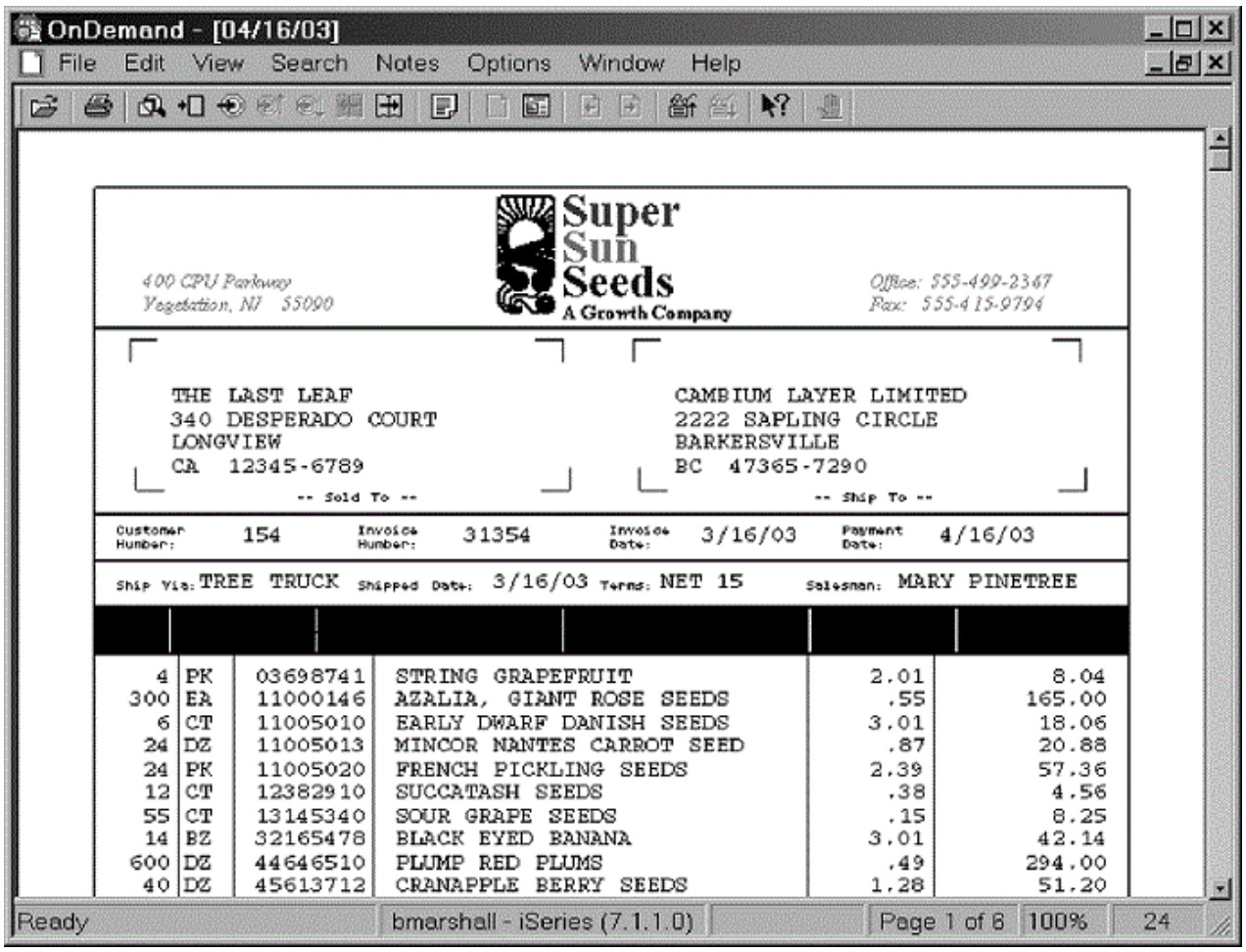
Multi-key index example

The following example demonstrates how to define a multi-key index by using the Report Wizard and the graphical indexer. The sample report to be archived is an AFP invoice. The following pieces of information should be used as indexes:

- Customer Number
- Invoice Number
- Invoice Date
- Item Number (this will be the multi-key index)
- Total Due

As a general rule, you should define triggers and fields from upper left to lower right of the report. This has the added benefit of making your indexer parameters easier to understand.

The following example shows a page from a sample report with a multi-key index.



1. First start the Content Manager OnDemand administrative client and log on to your instance's server.
2. Click the Report Wizard toolbar button. Then select the data type; for the example, select AFP. Then select the sample input file. The graphical indexer should now display the spooled file.

   The sample report contains AFP data, and only the text is displayed by the graphical indexer, not the AFP resources (such as special fonts, bar codes, graphics, and overlays).
3. Define the first trigger. Select the / (forward slash) character in the ship date as Trigger1. This trigger will be used to locate the Customer Number, Invoice Number, and Ship Date.
4. Define the second trigger. Select the . (period) character in the price as Trigger2. This trigger must be defined as a float trigger and will be used to locate the Item Numbers.
5. Define the third trigger. Select the / (forward slash) character in the payment due date as Trigger3. This trigger will be used to locate the Total Due.
6. After the triggers are defined, define the fields and indexes. When using the Report Wizard, the fields and indexes are defined in one step. If using the graphical indexer from within an application definition rather than the Report Wizard, the fields and indexes are defined in separate steps.

   The first field and index are for the customer number. The customer number is located by using Trigger1. On the Database Field Attributes page, the customer number field is defined as a string data type.

   The second field and index are for the invoice number. The invoice number is located by using Trigger1. On the Database Field Attributes page, invoice number is defined as a string data type.

   The third field and index are for the invoice date. The invoice date is located by using Trigger1. On the Database Field Attributes page, invoice date is defined as a date data type, and selected as our segment field.

The fourth field and index are for the item number. The item number is located by using Trigger2. On the Database Field Attributes page, item number is defined as a string data type.

The *Mask* parameter is used to specify a pattern that the field data must match in order to be used as an index. In the example, a field must consist of eight numeric characters (each # represents one numeric character). This could be useful if the trigger (a period) could be present in row that did not contain an item number.

After defining all of the fields, you must go back and mark the item number index as multi-key (as described in the following information).

The fifth field and index are for the total due. The total due is located by using Trigger3. On the Database Field Attributes page, total due is defined as a string data type.

That completes defining the fields and the indexes.

7. Now you must go back and specify the item number, which is Index4, as the multi-key. Click on the Toggle select Trigger, Index, Field Parameters toolbar button.

   The administrative client opens the Select dialog box.
8. Click on **Index 4**, and then click on the **Properties** button to open the Update an Index dialog box.
9. Click on the **Allow Multiple Values** check box, and click **OK** to save the item number index as a multi-key index.
10. Close the Select dialog box.
11. To verify how the system will index the document, click on the Toggle between Display and Add Parameters toolbar button.

    The defined triggers will be highlighted in red. The defined fields will be highlighted in blue.
12. You can now close the graphical indexer window and complete the process of using the Report Wizard to define the application group, application, and folder.

The following example shows the indexer parameters that were generated for the example report.

```
TRIGGER1=*,55,X'61',(TYPE=GROUP)                                          /* /       */
TRIGGER2=*,64,X'4B',(TYPE=FLOAT)                                          /* .       */
TRIGGER3=*,31,X'61',(TYPE=FLOAT)                                          /* /       */
FIELD1=0,15,6,(TRIGGER=1,BASE=0)
FIELD2=0,33,6,(TRIGGER=1,BASE=0)
FIELD3=0,50,8,(TRIGGER=1,BASE=0)
FIELD4=0,19,8,(TRIGGER=2,BASE=0,MASK='########')
FIELD5=0,69,12,(TRIGGER=3,BASE=0)
INDEX1=X'83A4A2A39596',FIELD1,(TYPE=GROUP,BREAK=YES)                      /* custno  */
INDEX2=X'8995A59596',FIELD2,(TYPE=GROUP,BREAK=YES)                        /* invno   */
INDEX3=X'8995A58481A385',FIELD3,(TYPE=GROUP,BREAK=YES)                    /* invdate */
INDEX4=X'89A3859495A494',FIELD4,(TYPE=GROUP,BREAK=NO,ALLOWMULTIPLEVALUES=YES)/* itemnum */
INDEX5=X'A396A3819384A485',FIELD5,(TYPE=GROUP,BREAK=NO)                   /* totaldue*/
```

After loading the example report, you can start the Content Manager OnDemand Client, open the new folder, and search for documents.

Defining transaction fields

A transaction report contains pages of records with one or more columns of sorted data. For example, each page of a general ledger report contains up to 80 transaction records. Each record contains a unique value, such as a transaction number. The records in the report are sorted on the transaction number.

Rather than storing every transaction number in the database (perhaps hundreds of thousands of rows), you can break the report into groups of pages (say, 100 pages in a group), extract the beginning and ending transaction number for each group of pages, and store the values in the database. Then, to retrieve the group of the report that contains a specific transaction number, a user specifies a transaction number. Content Manager OnDemand compares the transaction number with the beginning and ending values stored in the database and retrieves the group that matches the query.

To define a transaction report that contains one or more columns of sorted data as described in the example, a transaction field is used. A transaction field allows Content Manager OnDemand to index a group of pages using the first index value on the first page and the last index value on the last page.

The easiest method of specifying a transaction field is to use the Report Wizard and the graphical indexer.

The indexer parameter for the transaction field will look similar to the following:

```
FIELD1=*,*,10,(OFFSET=(3:12),MASK='##########',ORDER=BYCOL)
```

The indexer parameter for the index created from the transaction field will look similar to the following:

```
INDEX1=X'D3968195',FIELD1,(TYPE=GROUPRANGE,BREAK=NO)
```

These indexer parameters would be added by the Report Wizard to the Content Manager OnDemand Application definition. To see them, go to the Indexer Information tab, then click on Keyboard and then Modify to view the Application's Indexer Parameters.

Transaction report example

The following example demonstrates how to define a transaction report using the Report Wizard and the graphical indexer.

The sample report that we are archiving is a Loan Delinquency Report. Each page of the loan delinquency report contains loan records. Each record contains a unique value, the loan number. The records in the report are sorted on the loan number. We want to use the following pieces of information as indexes:

- Report Date
- Starting Page Number
- Loan Number (this will be the transaction field)

As a general rule you should define triggers and fields from upper left to lower right of the report. This has the added benefit of making your indexer parameters easier to understand.

The following example shows a sample page of the report.

```
REPORT   D33313001            ONDEMAND NATIONAL BANK                      DATE  01-15-00
BANK     001                                                              TIME  16:03:46
FROM     01/01/99                                                            MODE  9
TO       12/31/99               LOAN DELINQUENCY REPORT                      PAGE  0001

    LOAN        CUSTOMER             LOAN        DELINQUENT    DELINQUENT    DELINQUENT
   NUMBER         NAME              AMOUNT         30 DAYS       60 DAYS       90 DAYS

 0100000000  AARON, ROBERT         $10000000.00  $      50.00  $      50.00  $        .00
 0100000001  ABBOTT, DAVID         $   11000.00  $     100.00  $     200.00  $        .00
 0100000002  ABBOTT, DAVID         $   12000.00  $     140.00  $        .00  $        .00
 0100000003  ABBOTT, DAVID         $   13000.00  $     150.00  $        .00  $        .00
 0100000005  ROBINS, STEVEN        $     500.00  $      50.00  $        .00  $        .00
 0100000006  ARNOLD, SAMUEL        $    1000.00  $      75.00  $     150.00  $     225.00
 0100000007  PETERS, PAUL          $     650.00  $      50.00  $        .00  $        .00
 0100000008  ROBERTS, ABRAHAM      $    9000.00  $     120.00  $        .00  $        .00
 0100000009  SMITH, RANDOLPH       $    8000.00  $     115.00  $        .00  $        .00
 0100000010  KLINE, PETER          $    8500.00  $     110.00  $        .00  $        .00
```

To begin, first start the Content Manager OnDemand administrative client and log on to your instance's server. Next, click the Report Wizard toolbar button. Then select the data type; for the example, select SCS. Then select the sample input file. The graphical indexer should now display the spooled file.

Define the first trigger. Select the word REPORT for Trigger1. This trigger will be used to determine the start of the document, and to locate the Report Date and Starting Page Number fields.

Trigger1 is the only trigger required. Next, define fields and indexes. When using the report wizard, the fields and indexes are defined in one step. If using the graphical indexer within the application definition rather than the Report Wizard, the fields and indexes are defined in separate steps.

The first field and index are for the report date. The report date is located by using Trigger1. On the Database Field Attributes page, the report date is defined as a date data type and is selected as the segment field.

The second field and index are for the starting page number. The starting page number is located by using Trigger1. On the Database Field Attributes page, the starting page number is defined as an integer data type.

After defining all of the fields, you must change the starting page number field so that a new document group is not created each time the page number changes.

The third field and index are for the loan number. The loan number is located by using a mask. The Mask parameter is used to specify a pattern that the transaction field data must match in order to be used as an index. In the example, the field must consist of ten numeric characters (each # represents a numeric character). A transaction field does not use a trigger to locate the data, it uses the mask to define how the data must be structured, and uses any data on that page that matches that mask.

The Database Field Attributes page has specific parameters to support a transaction field. The end user of the sample report will see the folder field names. The database field names are used internally to Content Manager OnDemand and are not seen by end users.

The end user will enter search criteria (the loan number) into the field that is identified by the Query Folder Field. The document list will show two loan numbers. These are the starting and ending loan numbers of the group of the report that contains the loan number that was searched for.

The loan number is defined as a string data type.

Now you must go back and specify that the starting page number, which is Index2, should not start a new document group when the value changes. Click the Toggle select Trigger, Index, Field Parameters toolbar button.

The administrative client opens the Select dialog box.

Click on Index 2. Then click on the Properties button to open the Update an Index dialog box.

Under Break, select the No option. Click the OK button to save the starting page number index as a Break=No index. A change in the starting page number will no longer cause a new document group to be created.

Close the Select dialog box.

To verify how the system will index the document, click on the Toggle between Display and Add Parameters toolbar button.

The defined triggers will be highlighted in red. The defined fields will be highlighted in blue. The defined transactions fields will be highlighted in green.

You can now close the graphical indexer window and complete the process of using the Report Wizard to define the application group, application, and folder.

The following example shows the indexer parameters that were generated for the example report.

```
TRIGGER1=*,2,X'D9C5D7D6D9E3',(TYPE=GROUP)                          /* REPORT      */
FIELD1=0,83,8,(TRIGGER=1,BASE=0)
FIELD2=3,87,4,(TRIGGER=1,BASE=0)
FIELD3=*,*,10,(OFFSET=(3:12),MASK='##########',ORDER=BYROW)
INDEX1=X'998481A385',FIELD1,(TYPE=GROUP,BREAK=YES)                 /* rdate       */
INDEX2=X'A297818785',FIELD2,(TYPE=GROUP,BREAK=NO)                  /* spage       */
INDEX3=X'D396819540D5A494828599',FIELD3,(TYPE=GROUPRANGE,BREAK=NO) /* Loan Number */
```

After archiving the example report, you can start the Content Manager OnDemand client, open the new folder, and search for documents.

Understanding Translate Print Control

The need for the 400 indexer to interpret printer carriage control characters in the input data stream arises from the fact that some data can be very difficult to index if the number of lines of data varies greatly from document to document.

This often occurs with names and addresses, some of which may have one or two lines for the name and then one or two lines for the street address. Proper trigger values can be impossible to find, making indexing impossible. However, the solution is simple, since printer carriage control characters (such as a 0 (zero) for double space and a – (dash) for triple space) often cause the data to print consistently on the same lines from document to document, even if the data does not occur consistently on the same lines within the input.

Translate print control (TRANSLATEPRINTCONTROL=YES or NO) enables the indexing of reports in these circumstances. When translate print control is set to yes, the 400 indexer takes the input data for a page and rather than process it line by line, it reads the page buffer and creates a secondary view of the page. This secondary view of the page has each line of data placed as it would be on the printed page, after having interpreted (or translated) the carriage control characters.

The following table is a short example of what the raw and translated page buffers look like. The first column of the data field is the printer carriage control character.

Table 18. Raw and translated page buffers

Raw buffer	Translated buffer
Line Data 1 1 2 0 3 Bill Smith 4 0 123 First St. 5 0 Somewhere, NC 28105 6 22334455	Line Data 1 1 2 0 3 4 Bill Smith 5 0 123 First St. 6 7 0 Somewhere, NC 28105 8 9 22334455
Line Data 1 1 2 0 3 John Smith 4 456 Second Ave. 5 Appt 5C 6 0 Somewhere, NC 28105 7 23456677	Line Data 1 1 2 0 3 4 John Smith 5 456 Second Ave. 6 Appt 5C 7 0 Somewhere, NC 28105 8 9 23456677

Note that the TRANSLATEPRINTCONTROL parameter must be added manually to your indexer parameters by clicking the Keyboard radio button and then the Modify button in the Parameters Source area of the Indexer Information page of your Content Manager OnDemand Application definition.

Using system date or job run date as the value of a date field

If you need to create a date field in a Content Manager OnDemand application definition for a report that has no date printed on any of its pages, you can define the date field to use the system date at the time the report is loaded into Content Manager OnDemand or the run date of the job that loads the report.

Using the system date as the date value

If you wish to use the system date captured at the time the report is loaded into Content Manager OnDemand, do not specify a field or index in the Indexer Parameters, and simply enter '*t*' which indicates

the use of today's date for the Default Value of the date field on the Load Information tab of the Content Manager OnDemand application definition. This method will use the system date, not the job date, to determine the date value. This method works correctly even if you are using the output queue monitor, and continues to work correctly even if the monitor runs past midnight when the system date changes.

Note that simply using '*t*' as the DEFAULT= value for a date field is not allowed, such as:

```
FIELD1=0,95,8,(TRIGGER=1,BASE=0,DEFAULT=X'A3') /* t */
```

Using the job run date as the date value

If you are unable to simply use the system date for the date value, but instead need to set the date to a specific, "hard-coded" date (such as setting an annual report's date to January 1, no matter when it is actually loaded into Content Manager OnDemand), you can use the _*RUNDATE*_ special value on the FIELDx indexer parameter. For example, instead of using a FIELD1 definition like this which specifies a particular location of the date on the printed page of the report,

```
FIELD1=0,95,8,(TRIGGER=1,BASE=0)
```

you could use the hex equivalent of _*RUNDATE*_ for the field definition

```
FIELD1=X'6D5CD9E4D5C4C1E3C55C6D' /* _*RUNDATE*_ */
```

which causes the 400 indexer to use the job's run date instead.

To define your date field, you can either select Keyboard and then Modify from the Indexer Information page of the Content Manager OnDemand application definition and enter the FIELDx= indexer parameter manually, or you can use the graphical indexer and create it using the Add a Field panel.

You must also update the date field to use the %Y-%m-%d date format on the Load Information page of the Content Manager OnDemand application definition. The _*RUNDATE*_ special value returns the date in this format.

You must also add an index field. An example might look like this, to correspond to the FIELD1 shown earlier:

```
INDEX1=X'D9C4C1E3C5',FIELD1,(TYPE=GROUP,BREAK=NO) /* RDATE */
```

To define your date index, you can either select Keyboard and then Modify from the Indexer Information page of the Content Manager OnDemand application definition and enter the INDEXx= indexer parameter manually as shown in the preceding information, or you can use the graphical indexer and create it using the Add an Index panel.

When you are ready to load the report into Content Manager OnDemand, change your job's run date to the date you wish to have associated with the report, using the Date parameter of the Change Job command, for example `CHGJOB DATE(010108)` for January 1, 2008. Then, simply run the **ADDRPTOND** command interactively using the `ADDRPTOND SBMJOB(*NO)` command so that the interactive job's run date is used.

You can also use the Content Manager OnDemand output queue monitor to achieve the same results. First, start the monitor job using the **STRMONOND** command. If you need the date to be different than the current date, change the job date of the monitor job to the date you require. Then move the spooled files to the monitored output queue for processing. Only the spooled files with Content Manager OnDemand application definitions that use the _*RUNDATE*_ special value will use the monitor job's run date. Application definitions that explicitly define a date field from the print page will continue to extract the date from the report.

Note that the job date of the output queue monitor job does not automatically change at midnight. If you need the job date to change at midnight, you must use the CHGJOB command to explicitly set the new date, or end the monitor job before midnight and then restart it again after midnight to set the new date. Remember that the ending of the monitor could be done for you automatically if you specify an end time (on the Monitor Properties page of the IBM Content Manager OnDemand component of IBM Navigator for i, or the ENDTIME parameter of the **STRMONOND** command) when you start the monitor.

Alternative for creating sample data to define triggers, indexes, and fields

Define triggers, indexes, and fields from your actual data by using a graphical indexing tool in the Content Manager OnDemand Administrator client.

If the layout of your data varies between documents within your input file, you might be concerned about defining indexer parameters. These parameters successfully capture all the indexes when you use only the limited number of pages that are allowed for download to your workstation from your IBM i system by your graphics tool. Or you might need a use a different method to download your sample data to your workstation to define your triggers, indexes, and fields. You need this method if you do not have **IBM i Access Client Solutions** > **Windows Application Package** on your system.

Instead of using the Content Manager OnDemand Administrator client to download your sample data to your workstation, you can create sample data for the graphical indexer. You can create this sample data by using the `Print Text (PRTTXTOND)` command on your IBM i system.

1. To use this method, first locate the spooled file that you need to define to Content Manager OnDemand. For example, you can use the `Work with Spooled Files (WRKSPLF)` command on your IBM i system to find the spooled file.
2. On the **Work with All Spooled Files** window, use option 2 (Change) or option 8 (Attributes) to determine the spooled file name, job name, job user, job number, and spooled file number of the spooled file.
3. Enter the PRTTXTOND command and press **F4** to prompt for parameters before you press **Enter**.
4. Enter the spooled file attributes collected previously. Specify `*END` for the `Ending` page parameter to capture the entire spooled file.
5. Press **F10** for more parameters. For the Stream file (STMF) parameter, enter the IFS path and file name where you want the PRTTXTOND output stored.
6. After the PRTTXTOND command completes, map a drive from your workstation to the IBM i system.
7. Drag the sample data from the IBM i system to the directory on your workstation where you want to store the sample data. Alternately, you can use FTP to transfer the file from the IBM i system to your workstation.
8. After the file is on your workstation, open the Content Manager OnDemand Administrator client, logon to the server, and start the Content Manager OnDemand wizard by clicking the wizard icon.
9. After the Report wizard starts, select the data type, such as SCS or AFP, and click **Select Sample Data**.
10. On the Select File pane, click **PC File and Pages to Display** > **All Pages**.
11. Click **OK**, select the sample data file on your workstation, and click **Open**.
12. Markup the triggers, indexes, and fields for your site. The entire spooled file is available to help you verify the accuracy of your indexer parameters.

Defining text search fields

Important: This section details the standard text search function, not the full text search option that is available for purchase and provides a higher level of functioning.

The text search function is used to search for documents that contain a specified word or phrase that is not already defined as an index field for the documents. Initially, the specified index field values are used for the document search. Then, any document that matches the index fields criteria is searched for the specified text search word or phrase. For example, if the other index fields are date and account number, only documents that match the specified date and account number are searched for the specified text search word or phrase. Then, if a document contains the specified word or phrase, the document is added to the document list.

1. You can define only one text search field per folder.

2. The only valid search operator for a text search field is EQUAL.
3. Wildcards and pattern matching are not supported in a text search field.
4. The case of the specified word or phrase is ignored. For example, the phrase *customer xyz* matches *customer xyz*, *Customer Xyz*, and *CUSTOMER XYZ*.

The text search function is performed entirely on the IBM i server. Any performance impact will depend on the size and number of documents that are searched and on the performance of the system under the pre-existing workload. To limit the number of documents that are searched, users should specify criteria for some or all of the other index fields.

To create a text search field folder definition:

1. Create the application group, application, and folder by using the Report Wizard. (The Report Wizard does not include a provision for creating a text search field. However, doing so can be accomplished in just a few steps outside the Report Wizard.)
2. Copy the folder.
3. On the Field Definition tab, add a field named Full Text Search and select **Text Search** for the field type. Click the **Add** button to add the field.
4. Click **OK** to update the folder.

After archiving some documents into the application group, you can try the text search function.

You may want to set a number of options within the Content Manager OnDemand Windows client to enhance the use of text search:

- From the **Options** menu, select the **Show Search String** option. This option causes the text search string that you enter to be highlighted within the document after it is opened.
- If the `Autoview` option is set to either First Document or Single Document, the document automatically displays with the text search string highlighted. Single Document will cause the document to automatically display if only one document meets the search criteria. First Document always causes the first document in the document list to automatically display, not matter how many documents meet the search criteria.

When you are ready to try your text search field, open the folder that contains the text search field and perform a text search. The text search string can be one or more words. Open one of the documents from the document list. The text search string should be highlighted in the document. You can use the Find Next toolbar button to find the next occurrence of the string in the document. Note that you can still perform standard searches with the folder; you do not have to specify a text search every time that you search for documents.

To use the text search function with AFP or SCS-Extended documents, you must have the Portable Application Solutions Environment (PASE; a product option of IBM i) installed. If PASE is not installed, you will receive message 161 in the Content Manager OnDemand system log when attempting to perform a text search on AFP or SCS-Extended documents. To use the text search function with SCS or Line documents, you do not need PASE.

Chapter 5. PDF indexer

You can use the PDF indexer to extract index data from and generate index data about Adobe PDF files that you want to store in Content Manager OnDemand.

The index data can enhance your ability to store, retrieve, and view documents with Content Manager OnDemand. The PDF indexer 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.

Restriction: The PDF indexer is not supported on z/OS or Linux for System z platforms.

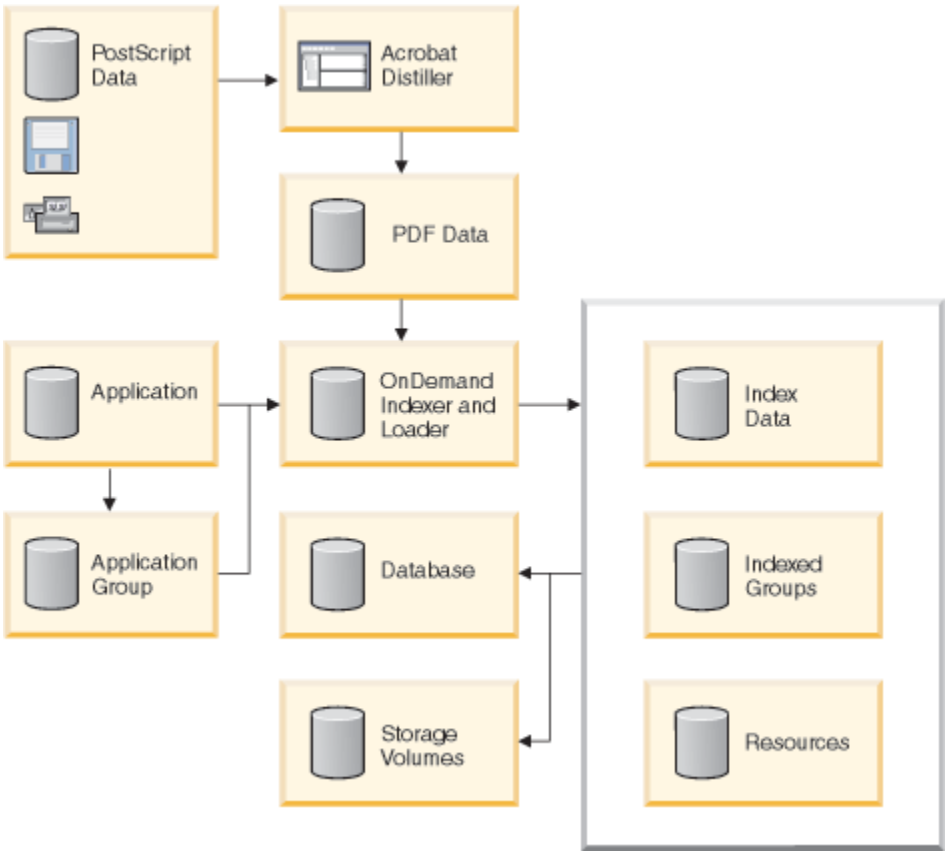
You process and store PDF documents on the server using standard Content Manager OnDemand functions. To process a document, you must define a Content Manager OnDemand application and application group. As part of the application, you must define the indexing parameters used by the PDF indexer to process input files.

You can automate the indexing and loading of data in Content Manager OnDemand for Multiplatforms by configuring and running the ARSLOAD program as a daemon (UNIX servers) or service (Windows servers).

In Content Manager OnDemand for i, you can automate the indexing and loading of data by using special parameters of the **ADDRPTOND** (using *STMF for the INPUT parameter) or **STRMONOND** (using *DIR for the TYPE parameter) commands or the ARSLOAD API program. See the API Reference section of the *IBM Content Manager OnDemand for i: Common Server Administration Guide* for more information on the ARSLOAD API program and its parameters.

After you index and store input files in Content Manager OnDemand, you can 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.

The following illustration shows the process of indexing and loading PDF input files.



The PDF indexer processes PDF input files. A PDF file is a distilled version of a PostScript file, adding structure and efficiency. A PDF file can be created by Acrobat Distiller or a special printer driver program called a PDFWriter. You can automate the distilling process by configuring and running the Distiller daemon (UNIX servers) or Acrobat Distiller (Windows servers). On IBM i, PDF stream files can be created by using IBM Transform Services for i, or by using InfoPrint Server. Content Manager OnDemand for i does not support processing PDF spooled files. 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.

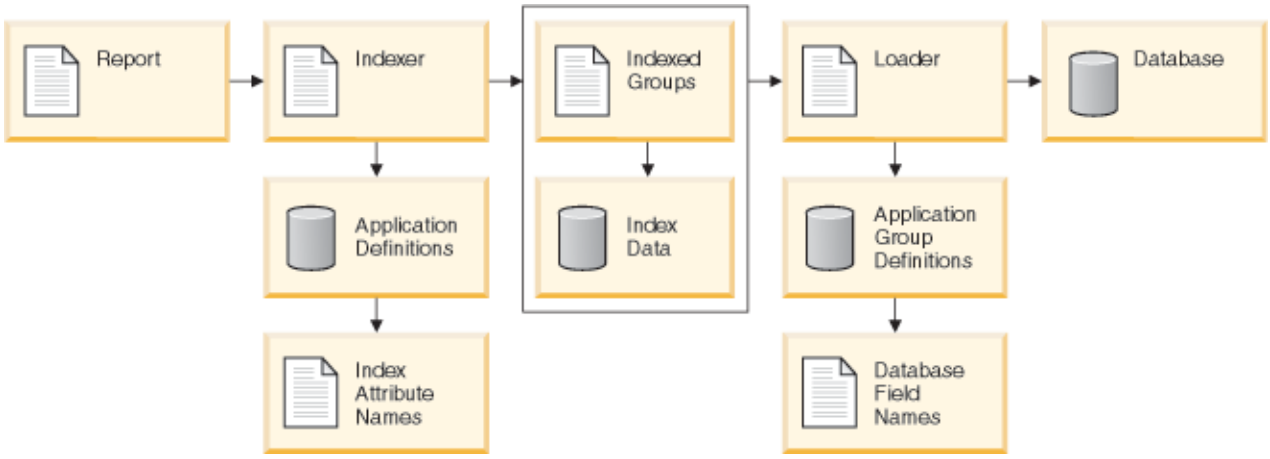
When the ARSLOAD program processes a PDF input file and the application Indexer Information tab specifies PDF as the indexer, it automatically calls the PDF indexer to process the input file. The PDF indexer processes the PDF input file with indexing parameters that determine the location and attributes of the index data. The PDF indexer extracts index data from the PDF file and generates an index file and an output file. The output file contains groups of indexed pages. A group of indexed pages can represent the entire input file or, more typically, one or more pages from the input file. If the input file contains logical groups of pages, such as statements or policies, the PDF indexer can create an indexed group for each statement or policy in the input file. 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 Content Manager OnDemand at the same time as the output file. After indexing the data, Content Manager OnDemand 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). On IBM i you can automate the process by using the STRMONOND *DIR command (recommended), or by running the ARSLOAD program as a daemon.

How Content Manager OnDemand processes index information

Content Manager OnDemand processes index information to help it complete several different types of tasks: loading data to the database, creating applications, creating application groups, and searching for reports in folders.

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 PDF indexer to process the indexing parameters and create the index data. The ARSLOAD program then loads the index data into the database, storing the group-level attribute values that the PDF indexing program extracted from the data into their corresponding database fields. The following illustration shows the index creation and data loading process.



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. Content Manager 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 Content Manager 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. Content Manager 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. Content Manager OnDemand lists the items that match the query. When the user selects an item for viewing, the Content Manager OnDemand client program retrieves the selected item from cache storage or archive storage.

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.

Restriction: The 32-bit OnDemand Administrator client fully supports working with PDF input files in the graphical indexer. PDF input files are not supported in the graphical indexer when using the 64-bit OnDemand Administrator client.

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 workstation from which you plan to run the OnDemand Administrator client.

Important: 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 C:\Program Files (x86)\IBM\OnDemand Clients\V10.5\PDF. The default Acrobat plug-in directory is C:\Program Files (x86)\Adobe\Acrobat *x.y*\Acrobat\plug_ins, where *x.y* is the version of Acrobat, such as 10.0 or 11.0 or 2016.

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 PDF indexer uses the triggers, fields, and indexes to locate the beginning of a document in the input data and extract index values from the input data. Once you have defined the triggers, fields, and indexes, you can save them in the application so that 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 PDF indexer attempts to locate the specified string in the specified location. When a match occurs, the PDF indexer knows that it has found the beginning of a document. The fields and indexes are based on the location of the trigger.

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

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

1. To begin, start the administrative client.
2. Log on to a server.
3. Start the report wizard by clicking the Report Wizard button on the toolbar. The report wizard opens the Sample Data dialog box.
4. Click Select Sample Data to open the Open dialog box.

   Restriction: For IBM i users: The PDF Indexer can process only stream files when running on IBM i. PDF spooled files are not supported.
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.
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.

Indexing input data

Indexing parameters include information that allow the PDF indexer to identify key items in the print data stream, *tag* these items, and create *index elements* pointing to the tagged items.

Content Manager OnDemand uses the tag and index data for efficient, structured search and retrieval. You specify the index information that allows the PDF indexer to segment the data stream into individual items, called *groups*. A group is a collection of one or more pages, such as a bank statement, insurance policy, phone bill, or other logical segment of a report. The PDF indexer creates indexes for each group when the value of an index changes (for example, account number).

A tag is made up of an *attribute name*, for example, Customer Name, and an *attribute value*, for example, Earl Hawkins. Tags also include information that tell the PDF indexer where to locate the attribute value on a page. For example, a tag used to collect customer name index values provides the PDF indexer with the starting and ending position on the page where the customer name index values appear. The PDF indexer generates index data and stores it in a generic index file.

Related reference

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.

Coordinate system

The location of the text strings the PDF indexer uses to determine the beginning of a group and index values are described as *x* and *y* pairs in a coordinate system imposed on the page.

For each text string, you identify its upper left and lower right position on the page. The upper left corner and lower right corner form a string box. The string box is the smallest rectangle that completely encloses the text string. The origin is in the upper left hand corner of the page. The *x* coordinate increases to the right and *y* increases down the page. You also identify the page on which the text string appears. For example, the text string Customer Name, that starts 4 inches to the right and 1 inch down and ends 5.5 inches to the right and 1.5 inches down on the first page in the input file can be located as follows:

```
ul(4,1),lr(5.5,1.5),1,'Customer Name'
```

The ARSPDUMP command is used to identify the locations of text strings on the page.

Remember: The ARSPDUMP command is not supported on z/OS or Linux for System z platforms..

Indexing parameters

Processing parameters can contain index and conversion parameters, options, and values. For most reports, the PDF indexer requires at least three indexing parameters to generate index data.

TRIGGER

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

The PDF indexer supports the following types of triggers:

- GROUP TRIGGERS

  The PDF indexer compares words in the input file with the text string specified in a trigger. The location of the trigger string value must be identified using the x,y coordinate system and page offsets.

  A maximum of 16 triggers (group or float) can be specified.

  Group triggers are used in conjunction. For example, all the group triggers must match before the PDF indexer can begin to locate index information.

The group triggers and the fields based on them are used to define the extent of the groups.

The indexer must find all the group triggers at least once within the document or it will stop processing and issue an error message.

- FLOAT TRIGGERS

  Float triggers are used to locate fields which might occur more than once within a group, or might not occur at all. The PDF indexer compares words in the input file with the text string specified in a trigger. The location of the trigger string value must be identified using the x,y coordinate system and page offsets.

  A maximum of 16 triggers (group or float) can be specified.

  The float trigger must match before the PDF indexer can begin to locate index information. The fields based on floating triggers do not define the extent of the groups.

  If a floating trigger is not found, the indexer continues processing with no error.

  The following rules apply when using floating triggers:

  1. Trigger1 must be a group trigger.
  2. Fields based on floating trigger must contain a default value.
  3. Fields based on floating triggers cannot be combined with any other field in an index.
  4. At least one index must contain a field (or fields) based on a group trigger.

FIELD

The field parameter specifies the location of the data that the PDF indexer uses to create index values.

- Field definitions are based on TRIGGER1 by default, but can be based on any of 16 TRIGGER parameters.
- The location of the field must be identified using the *x,y* coordinate system and page offsets.
- A maximum of 128 fields can be defined.
- A field parameter can also specify all or part of the actual index value stored in the database.

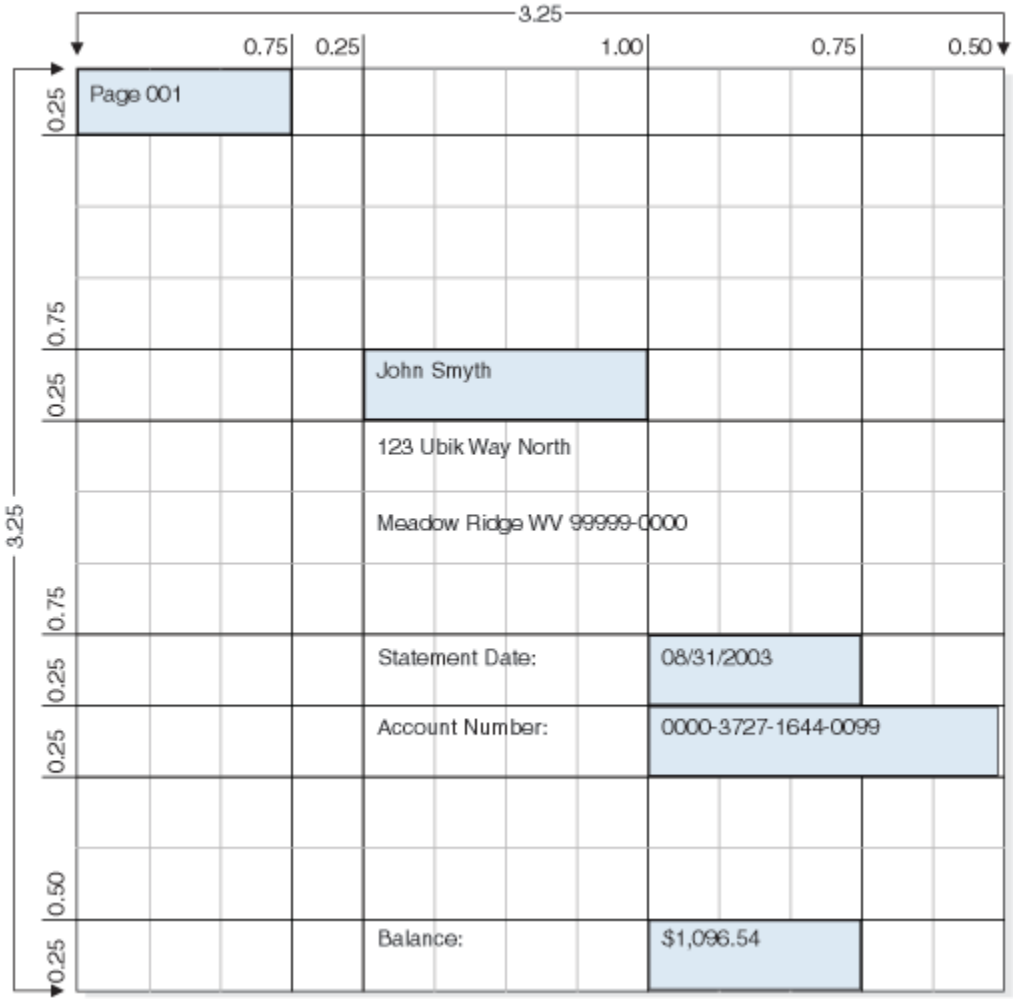
INDEX

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

- The PDF indexer creates indexes for a group of one or more pages.
- You can concatenate field parameters to form an index, unless any of the fields was based on a floating trigger. Fields based on floating triggers cannot be combined with any other field in an index.
- A maximum of 128 index parameters can be specified.

The PDF indexer creates a new group and extracts new index values when one or more of the index values change, unless the index contains a field based on a floating trigger. Fields based on floating triggers cannot be used to create a new group.

The following example 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.



TRIGGER parameters tell the PDF indexer how to identify the beginning of a group in the input. The PDF indexer requires one TRIGGER parameter to identify the beginning of a group (statement) in the sample file. FIELD parameters determine the location of index values in a statement. Fields are based on the location of trigger records. INDEX parameters identify the attribute names of the index fields. Indexes are based on one or more field parameters. The following parameters could be used to index the report:

- Define a trigger to search each page in the input data for the text string that identifies the start of a group (statement):

```
TRIGGER1=ul(0,0),lr(.75,.25),*,'Page 001'
```

- Define fields to identify the location of index data. For the sample report, you might define four fields:
  - FIELD1 identifies the location of customer name index values.

    ```
    FIELD1=ul(1,1),lr(2,1.25),0
    ```

  - FIELD2 identifies the location of statement date index values.

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

  - FIELD3 identifies the location of account number index values.

    ```
    FIELD3=ul(2,2.25),lr(3.25,2.5),0
    ```

  - FIELD4 identifies the location of the balance index values.

    ```
    FIELD4=ul(2,3),lr(2.75,3.25),0
    ```

- Define indexes to identify the attribute name for an index value and the field parameter used to locate the index value.
  - INDEX1 identifies the customer name, for values extracted using FIELD1.

    ```
    INDEX1='cust_name',FIELD1
    ```

  - INDEX2 identifies the statement date, for values extracted using FIELD2.

    ```
    INDEX2='sdate',FIELD2
    ```

  - INDEX3 identifies the account number, for values extracted using FIELD3.

    ```
    INDEX3='acct_num',FIELD3
    ```

  - INDEX4 identifies the balance, for values extracted using FIELD4.

    ```
    INDEX4='balance',FIELD4
    ```

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 most current PDF Reference, section 10.2.

When INDEXMODE=METADATA is specified, the PDF indexer extracts fields from the Document Information Dictionary that correspond to the following metadata keywords, if they exist, and places their values into the `.ind` file:

- Title
- Author
- Subject
- 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 PDF indexer 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 PDF indexer converts dates that are specified in the PDF format of (D:*YYYYMMDDHHmmSSOHH'mm*) to a format of *YYYYMMDDHHmmSS*. The index values CreationDate and ModDate contain the date formatted with the local time. If the time zone information is specified in the PDF date (the OHH'mm section) the PDF indexer creates another index value named CreationDateTZ or ModDateTZ which contains the date formatted with the time adjusted to Universal Time. For more information on Adobe date formats, see the most current PDF Reference, section 3.8.3.

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 Index File Format
COMMENT:
COMMENT:
COMMENT:Code Page of the Index Data
CODEPAGE:1208
COMMENT:Index Field(s)
GROUP_FIELD_NAME:Title
GROUP_FIELD_VALUE:Administrator's Guide
GROUP_FIELD_NAME:Author
GROUP_FIELD_VALUE:IBM
GROUP_FIELD_NAME:Creator
GROUP_FIELD_VALUE:XPP
GROUP_FIELD_NAME:Producer
GROUP_FIELD_VALUE:IBM
GROUP_FIELD_NAME:CreationDate
GROUP_FIELD_VALUE:20090408173745
GROUP_FIELD_NAME:CreationDateTZ
GROUP_FIELD_VALUE:20090408233745
GROUP_FIELD_NAME:ModDate
GROUP_FIELD_VALUE:20090408173745
GROUP_FIELD_NAME:ModDateTZ
GROUP_FIELD_VALUE:20090408233745
COMMENT:Index Offsets and Length
GROUP_OFFSET:0
GROUP_LENGTH:748641
GROUP_PAGES:387
GROUP_FILENAME:\pdf\pdfoutput\admin.pdf
COMMENT:
COMMENT:
COMMENT:
COMMENT:End Generic Indexing File
```

Indexing with internal indexes

PDF internal indexes are contained inside the PDF document, similar to the way that TLEs are contained inside an AFP document. These indexes are not part of the viewable page, but they can be extracted by the PDF Indexer and placed into the index file.

Extracting the internal indexes can give better performance during loading than searching for and extracting the indexes from the pages of a PDF document.

PDF internal indexes must be created by the PDF provider when the document is created, in the same way that TLEs must be created in an AFP document at creation time.

The internal indexes are contained within the Page-Piece Dictionary, which is an optional structure of PDF document architecture. See the Adobe PDF Reference 1.7, section 14.5 for a technical description of the Page-Piece Dictionary. Each page of a PDF document may contain a Page-Piece Dictionary.

In order for the PDF Indexer to be able to extract the indexes, the Page-Piece Dictionary must be named IBM-ODIndexes. This name is case-sensitive.

The first group begins with the page that contains the first Page-Piece Dictionary. Pages before the first occurrence are discarded. When any of the index values in the Page-Piece Dictionary changes, a new group is started. Each Page-Piece Dictionary must contain the same number of indexes.

It is not necessary for every page to contain a Page-Piece Dictionary. If a page does not contain one, that page is associated with the previous group.

There are PDF generators that you can use to create a Page-Piece Dictionary using a graphical interface. The following is an example of how a Page-Piece Dictionary would appear inside a PDF document. The first index has an index name of "DocId" and the value is "AAA". The LastModified date is required by the PDF architecture.

```
/PieceInfo <</IBM-ODIndexes <</Private
                              <</DocId(AAA)
```

```
                                  /BankNumber(0000000001)
                                  /AcctNumber(00000000000111111111)
                                  /NoticeType(W)
                                  /StmtDate(20120507)
                                 >>
                                /LastModified(D:20120619000000Z)
                               >>
            >>
```

The values must be encoded in code page 1208. To specify a value in hexadecimal, enclose the value in angle brackets. For example, the previous "DocId" value specified in hexadecimal would appear as follows:

```
/DocId<414141>
```

Use the following indexing parameters to extract internal indexes. No other parameters are necessary.

```
INDEXSTARTBY=1
RESTYPE=ALL
INDEXMODE=INTERNAL
```

The INDEXSTARTBY parameter determines the page number by which the PDF indexer must locate the internal indexes for the first group (document) within the input file. The PDF indexer stops processing if it does not locate the internal indexes for the first group by the specified page number.

The default number of indexes supported is 32. For more indexes, specify INDEXMODE=INTERNAL,n where n is the number of indexes. To conserve memory, use the minimum number required. For example, for the Page-Piece Dictionary shown in the preceding example, use INDEXMODE=INTERNAL,5.

The PDF Indexer can extract indexes using TRIGGER, FIELD and INDEX parameters, or it can extract indexes from the Page-Piece Dictionary with the INDEXMODE=INTERNAL parameter, but it cannot use both methods in the same file.

The PDF Indexer supports up to 128 internal indexes.

Multiple Page-Piece Dictionary rows

The typical use for a Page-Piece Dictionary is to create one row of indexing information per group. However, starting with version 9.5.0.8, it is possible to create multiple index rows per group using a Page-Piece Dictionary. The following explains the Page-Piece Dictionary format that the PDF indexer requires in order to process multiple index rows for a group.

Use keys IBM-ODIndexes, IBM-ODIndexes1, IBM-ODIndexes2, ...IBM-ODIndexesn to designate each set of indexes on a page. A page can contain one, many, or no Page-Piece Dictionaries.

For example:

```
/PieceInfo<</IBM-ODIndexes<</LastModified(D:19700101000000Z)
                             /Private<<
                                     /AdvType(DIVIDEND WITH REINVEST OPTION)
                                     /CIFNo(5419-1)
                                     /ExDate(03 AUG 2004)
            /IBM-ODIndexes1<</LastModified(D:19700101000000Z)
                             /Private<<
                                     /AdvType(DIVIDEND WITH REINVEST OPTION)
                                     /CIFNo(5419-2)
                                     /ExDate(03 AUG 2004)
            /IBM-ODIndexes2<</LastModified(D:19700101000000Z)
                             /Private<<
                                     /AdvType(DIVIDEND WITH REINVEST OPTION)
                                     /CIFNo(5419-3)
                                     /ExDate(03 AUG 2004)>>>>
```

The example will create three rows pointing to the same document:

```
DIVIDEND WITH REINVEST OPTION   5419-1   03 AUG 2004
DIVIDEND WITH REINVEST OPTION   5419-2   03 AUG 2004
DIVIDEND WITH REINVEST OPTION   5419-3   03 AUG 2004
```

Rules for multiple Page-Piece Dictionaries:

1. A page with no Page-Piece Dictionaries is part of the current group.
2. A page with multiple Page-Piece Dictionaries starts a new group.
3. If the current group contains multiple Page-Piece Dictionaries and a page is encountered that contains only one Page-Piece Dictionary, it starts a new group.
4. If the current group contains one Page-Piece Dictionaries and a page is encountered that also contains one Page-Piece Dictionary, the index values are compared and if any value changes, a new group is started. This is the default behavior.
5. Each `IBM-ODIndexes` entry must contain the same number of indexes.
6. A page can contain up to 10,000 Page-Piece Dictionaries.

Using regular expressions

A regular expression is a pattern which is used to match characters in a string.

There are many excellent online resources which explain the syntax rules of regular expressions. The following are examples of some of the most common:

A character string, for example “Account” will look for the characters "Account". By default searches are case sensitive.

- [A-Z] Look for one uppercase letter.
- [A-Z]{3} Look for three consecutive uppercase letters.
- [0-9]{5} Look for five consecutive digits.
- [0-9]+ Look for one or more digits.
- [^a-z] Look for everything except lowercase a to z.
- \s (Lowercase s) Look for one whitespace character (space, tab, etc).
- \S (Uppercase S) Look for any character not whitespace.

The PDF indexer can use a regular expression in the **TRIGGER** and **FIELD** parameter. In the **TRIGGER**, the regular expression specifies the pattern for which to search; in the **FIELD**, the regular expression is applied to the characters which have been extracted from the field in a way similar to using a mask.

Here is an example:

```
TRIGGER1=UL(1.00,3.89),LR(2.52,4.17),*,REGEX='PAGE 1'
TRIGGER2=UL(1.02,4.60),LR(2.11,4.95),0,REGEX='[0-9]{5} [a-z]{4}'
FIELD1=UL(1.44,0.00),LR(2.75,0.30),0,(TRIGGER=2,BASE=TRIGGER,
REGEX='[A-Z]+ [A-Z] [A-Z]+')
INDEX1='Name',FIELD1,(TYPE=GROUP)
```

In this example TRIGGER1 uses a regular expression specified as an ordinary text string. TRIGGER2 uses a regular expression which specifies a pattern of five digits, followed by a space, followed by four lowercase letters. The text "12345 acct" would match the pattern.

FIELD1 uses a regular expression, which specifies one or more uppercase letters, followed by a space, followed by a single uppercase letter, followed by a space, followed by one or more uppercase letters. The characters "MARY R SMITH", "W A DOE", or "LARRY G W" would match this regular expression.

Using a regular expression on the TRIGGER parameter

On the TRIGGER parameter use the regular expression instead of a value. A regular expression can be used on both a group trigger and a floating trigger.

The maximum length of the regular expression is 254 characters.

In order to find the text which matches the regular expression, the PDF indexer creates a string containing all the text within the trigger bounding box. Each word in the string is separated by one space. It then applies the regular expression to the string.

The maximum string length to which the regular expression can be applied is 2000 characters. If the text within the bounding box is longer, the PDF indexer ends with the error message ARS4948 “Regular expression buffer exceeded.”

Once the regular expression matches text in the text string, the PDF indexer looks for the next trigger, or, if all the group triggers have been found, it collects the fields.

Using a regular expression on the FIELD parameter

On the FIELD parameter use the regular expression instead of a mask. A mask and a regular expression cannot both be specified on the same FIELD parameter. The regular expression can be specified on a field based on either a group trigger or a floating trigger.

The maximum length of the regular expression is 2000 characters.

In order to find the text which matches the regular expression, the PDF Indexer creates a string containing all the text within the field bounding box. Each word in the string is separated by one space. It then applies the regular expression to the string. Any text that matches the regular expression is extracted for the field.

Considerations

- Performance using a regular expression may not be as fast as using a value.
- If the regular expression is invalid, the PDF indexer will fail with error message ARS4950 “Invalid regular expression.”

Using Default Values

If the regular expression does not match any text in the field, the default value specified on the FIELD parameter is used. If no default value is specified, the PDF Indexer ends with error message APK4915 “Field x not found on page y.” Default values are required for fields based on floating triggers.

The load process uses the default value in the Application when a floating trigger is not found within a group. Since the trigger is not found, there is no field for that group.

Examples

Using a regular expression for a trigger:

```
TRIGGER1=UL(1.00,3.89),LR(2.52,4.17),*,REGEX='P[A-Z]{3} '
```

This regular expression will match text that begins with the letter 'P' and is followed by three uppercase letters followed by a space, for example, "PAGE ".

Using a regular expression to extract a date from a field in the form of "July 4, 1956":

```
FIELD1=UL(0.54,0.40),LR(1.64,0.67),0,(TRIGGER=1,BASE=0,
REGEX='[A-Z][a-z]+ [0-9]+, [0-9]{4}',DEFAULT='January 1, 1970')
INDEX1='RDate',FIELD1
```

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 PDF indexer 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.

Remember: The ARSPDUMP command is not supported on z/OS or Linux for System z platforms.

PDF fonts and output file size

The fonts that are used in a PDF document are one of the factors that determines the indexing's output file size.

The base 14 Type 1 fonts
For every PDF data stream, there exists a core set of fonts that are ensured to be available to the Acrobat program. Because they are available on the system, they are not embedded in the document. Therefore, documents that are created with these fonts are more compact. These fonts are known as the *base 14 fonts*:

- Courier
- Courier-Bold
- Courier-BoldOblique
- Courier-Oblique
- Helvetica
- Helvetica-Bold
- Helvetica-BoldOblique
- Helvetica-Oblique
- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Symbol
- ZapfDingbats

Fonts that are not members of the base 14 fonts might be embedded in the document, or they might be stored in a font directory. Images and bar code fonts are also embedded in the document.

The PDF Indexer collects resources such as fonts and images, removes them from the document, and places them in a resource file. The number of embedded fonts in the document directly affects the size of the resource file.

You should use only the base 14 fonts when you create PDF documents. Because these fonts are not embedded in the document, documents that are created with these fonts are smaller, and the resource file is also smaller.

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 PDF indexer 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 Adobe PDF reference. These fonts are guaranteed to be available on the client, therefore, they are not embedded in the report.

The resources that the PDF indexer collects are based on the value of the RESTYPE parameter. The following table lists values for this parameter.

Table 19. 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 PDF indexer.

PDF indexing system requirements

The requirements for running Content Manager OnDemand are published on the IBM support site.

Related information

Content Manager OnDemand for Multiplatforms Version 10.5 requirements

Content Manager OnDemand for i Version 7.4 requirements

Specifying the location of Adobe® fonts

If a font is referenced in an input file but not embedded in the file or is not one of the 14 base fonts, and the PDF indexer cannot locate the font, the indexing will fail. If you purchase additional fonts and install

them on the system, the additional fonts can be found at indexing time by specifying the location with the **FONTLIB** parameter.

IBM i only: If you installed fonts for use with the PDF indexer, you should verify the location of the fonts. Fonts can be located in any directory, but you must specify the directory using the **FONTLIB** parameter. For example:

```
FONTLIB=/QIBM/ProdData/OnDemand/Adobe/Resource/CMap
```

PDF indexing limitations

You can use the PDF indexer to generate index data for PostScript and PDF files that are created by user-defined programs.

Remember:

- The PDF indexer can process input files that are up to 4 GB in size.
- The PDF indexer supports DBCS languages. However, IBM does not provide any DBCS fonts. You can purchase DBCS fonts from Adobe. The PDF indexer supports all DBCS fonts, except encrypted Japanese fonts.
- Input data delimited with PostScript Passthrough markers cannot be indexed
- The Adobe Toolkit does not validate link destinations or bookmarks to other pages in a document or to other documents. Links or bookmarks may or may not resolve correctly, depending on how you segment your documents.
- Multiplatform users: 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 or another application to convert the PDF into a data stream suitable for printing.
- IBM i users: Server printing of PDF documents is not supported on IBM i. Instead, you must print from the client.
- If a font is referenced in an input file but not embedded in the file or is not one of the 14 base fonts, and the PDF indexer cannot locate the font, the indexing will fail.
- The PDF indexer does not support documents containing Digital Signatures, or that are password protected.

The PDF Indexer was tested using documents containing up to 100,000 pages. However, there are many factors that affect the number of pages that can be successfully indexed and stored on your system. Those factors include:

- the system resources available such as CPU, memory, and disk.
- the size the PDF input file.
- the type and number of resources such as fonts and images used in the PDF input file.

If your PDF file does not store successfully, consider:

- splitting the file into a number of separate, smaller files.
- reducing the number of different fonts used.
- changing the type of fonts used.
- reducing the number or size of the images included in the file.

Input data requirements

The PDF indexer processes PDF input data.

Multiplatform input data requirements

PostScript data generated by applications must be processed by Acrobat Distiller before you run the PDF indexer.

The online documentation provided with Acrobat Distiller describes methods you can use to generate PDF data.

You can use several methods to provide the PDF indexer with access to input data, including FTP and NFS. If you use a file transfer method to copy PDF data to the 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.YYDDD.HHMMSST.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.

IBM i input data requirements

The PDF Indexer processes PDF input data. The Content Manager OnDemand directory monitor (started with the STRMONOND command with *DIR specified for the Type parameter) and the ADDRPTOND command are the two most common ways to invoke the PDF Indexer to index and load PDF data into Content Manager OnDemand on IBM i. You can also use the ARSLOAD API.

The PDF Indexer generates the index data and then adds the index information to the database and loads the input data on to the storage media defined for the particular Content Manager OnDemand application group to which the data belongs.

If you plan to automate the data indexing and loading process on the Content Manager OnDemand server, either the input file name, specific parameters on the command used to load the data, or a monitor user exit program must identify the application group and application to load. The PDF file name extension is required to initiate a load process. The case (uppercase or lowercase) of the extension (.pdf) is ignored. Application group and application names are case sensitive. Application group and application names may include special characters such as the blank character when using ADDRPTOND or ARSLOAD with a specific application group and application name provided. However, STRMONOND and ARSLOAD when using the MVS naming convention (-A and -G parameters) do not support archiving PDF files that have spaces in the file name. See the *IBM Content Manager OnDemand for i: Common Server Administration Guide* for more information about using the STRMONOND and ADDRPTOND commands and the ARSLOAD API to load data into Content Manager OnDemand.

National language support for indexed PDF documents

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

See "Specifying the location of Adobe® fonts" on page 223 if you plan to use DBCS font files.

Data values that you specify on TRIGGER and FIELD parameters (either as plain text or in hexadecimal) must be encoded in UTF-8. These data values include trigger values, field default and constant values, and index names.

On Windows environments, set the system locale in the Region and Language dialogue.

Attention: Plain text in the IBM i environment is UTF-8.

For more information about NLS in Content Manager OnDemand, see the *IBM Content Manager OnDemand for Multiplatforms: Installation and Configuration Guide* or *IBM Content Manager OnDemand for i: Common Server Planning and Installation Guide*.

PDF indexer parameters

This parameter reference assumes that multiplatform users will use the ARSLOAD program to process your input files. If you are using IBM Content Manager OnDemand for i, this reference assumes you will use the **STRMONOND** command, **ADDRPTOND** command, or ARSLOAD API to process your input files.

When you use these methods to process input files, the PDF indexer ignores any values that you may provide for the INDEXDD, INPUTDD, MSGDD, OUTPUTDD, and PARMDD parameters (and RESOBJDD for IBM i users).

If you run the ARSPDOCI program or API 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. The ARSPDOCI program is not supported on z/OS .

If you must include spaces in a value for an option of a PDF Indexer parameter, enclose the entire value in quotation marks.

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. 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=[YES | NO]

Options and values

The *value* can be:

YES
The bookmarks are copied to each new document that is created by the PDF indexer. This is the default value.

NO
The bookmarks are not copied to new documents.

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.

DISABLECHARREORDERING

The character reordering affects how text is extracted from PDF documents. By default, the character reordering is disabled. However, if the PDF page contains heavily overlapped characters, enabling the character reordering by setting *DISABLECHARREORDERING = 0* may produce more consistent text extraction.

Required?
No

Default Value
1

Syntax

DISABLECHARREORDERING=*value*

Options and values

The *value* can be:

0
Character reordering is not disabled. Character reordering might occur during the processing of the PDF input file.

1
Character reordering is disabled and will not occur (the default).

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 128 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*'| **REGEX=**'*regular_expression*',**DEFAULT=**'*value*')]

Trigger field options and values

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

ul(*x,y*)
The coordinates for the upper left corner of the field string box. The field string box is the smallest rectangle that completely encloses the field string value (one or more words on the page). The PDF indexer must find the field string value inside the field string box. The supported range of values is 0 (zero) to 200, page width and length, in inches.

lr(*x,y*)
The coordinates for the lower right corner of the field string box. The field string box is the smallest rectangle that completely encloses the field string value (one or more words on the page). The PDF indexer must find the field string value inside the field string box. The supported range of values is 0 (zero) to 200, page width and length, in inches.

page
The sheet number where the PDF indexer begins searching for the field, relative to a trigger or 0 (zero) for the same page as the trigger. If you specify BASE=0, the *page* value can be –16 to 16. If you specify BASE=TRIGGER, the *page* value must be 0 (zero), which is relative to the sheet number where the trigger string value is located.

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

BASE={0|TRIGGER}
Determines whether the PDF indexer uses the upper left coordinates of the trigger string box to locate the field. Choose from 0 (zero) or TRIGGER. If BASE=0, the PDF indexer adds zero to the field string box coordinates. If BASE=TRIGGER, the PDF indexer adds the upper left coordinates of the location of the trigger string box to the coordinates provided for the field string box. This is an optional keyword, but the default is BASE=0.

You should use BASE=0 if the field data always starts in a specific area on the page. You should use BASE=TRIGGER if the field is not always located in the same area on every page, but is always located a specific distance from a trigger. This capability is useful when the number of lines on a page varies, causing the location of field values to change. For example, given the following parameters:

```
TRIGGER2=ul(4,4),lr(5,8),1,'Total'
FIELD2=ul(1,0),lr(2,1),0,(TRIGGER=2,BASE=TRIGGER)
```

The trigger string value can be found in a one by four inch rectangle. The PDF indexer always locates the field in a one inch box, one inch to the right of the location of the trigger string value. If the PDF indexer finds the trigger string value in location `ul(4,4),lr(5,5)`, it attempts to find the field in location `ul(5,4),lr(6,5)`. If the PDF indexer finds the trigger string value in location `ul(4,6),lr(5,7)`, it attempts to find the field in location `ul(5,6),lr(6,7)`.

Important: 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 <= 200 and 0 <= y <= 200. The `ul(x,y)` and `lr(x,y)` coordinates of the FIELD parameter are relative offsets from the `ul(x,y)` coordinates of the trigger. For example, suppose the field string rectangle is located at `ul(1,1), lr(2,2)` which is an absolute location on the page. If the trigger string rectangle is located at `ul(5,5), lr(7,7)`, then the field coordinates would be `ul(-4,-4), lr(-3,-3)`.

MASK='*field_mask*'
The pattern of symbols that the PDF indexer matches to data located in the field. When you define a field that includes a mask, an **INDEX** parameter based on the field cannot reference any other fields. A mask and a regular expression cannot both be specified on the same **FIELD** parameter. Valid mask symbols can include:

@
Matches alphabetic characters. For example:

```
MASK='@@@@@@@@@@@@@@@'
```

Causes the PDF indexer to match a 15-character alphabetic field, such as a name.

#
Matches numeric characters. For example:

```
MASK='##########'
```

Causes the PDF indexer to match a 10-character numeric field, such as an account number.

¬
Matches any non-blank character.

^
Matches any non-blank character.

%
Matches the blank character and numeric characters.

=
Matches any character.

Important: The string that you specify for the mask can contain any character. For example, given the following definitions:

```
FIELD2=UL(0.46,3.47),LR(0.82,7.46),0,(TRIGGER=2,BASE=0,MASK='@000-####-#')
```

The IBM Content Manager OnDemand PDF indexer selects the field only if the data in the field 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.

REGEX='*regular_expression*'
The regular expression that the IBM Content Manager OnDemand PDF indexer matches to data located in the field. Either **MASK** or **REGEX** can be specified, but not both. The maximum length of the regular expression is 2000 characters. For more information see "Using regular expressions" on page 220.

Note: The string that you specify for the regular expression can be any valid regular expression. For example, given the following definition:

```
FIELD2=UL(0.46,3.47),LR(0.82,7.46),0,(TRIGGER=2,BASE=0,
  REGEX='[A-Z][0]{3}-[0-9]{4}-[0-9]')
```

The PDF indexer selects the field only if the data in the field contains an eleven-character string comprised of (in order) any uppercase 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, or if a mask or regular expression does not match any characters within the bounding box. A field that is based on a floating trigger must contain a default value. You can specify the default value in hexadecimal.

Attention: If you specify the value in hexadecimal, it must be specified in UTF-8.

For example, assume that an application program generates statements that contain an audit field. The contents of the field can be PASSED or FAILED. However, if a statement has not been audited, the application program does not generate a value. In that case, there are no words within the field string box. To store a default value in the database for unaudited records, define the field as follows:

```
FIELD3=ul(8,1),lr(8.5,1.25),1,(DEFAULT='NOT AUDITED')
```

The PDF indexer assigns the index associated with FIELD3 the value NOT AUDITED, if the field string box is blank.

Trigger field examples

The following field parameter causes the PDF indexer to locate the field at the coordinates provided for the field string box. The field is based on TRIGGER1 and located on the same page as TRIGGER1. Specify BASE=0 because the field string box always appears in a specific location on the page.

```
TRIGGER1=ul(0,0),lr(.75,.25),*,'Page 0001'
FIELD1=ul(1,1),lr(3.25,1.25),0,(TRIGGER=1,BASE=0)
```

Hexadecimal default value:

```
TRIGGER1 = ul(4.5,1.25),lr(5.75,1.5), *,'ACCOUNT'
FIELD1   = ul(6.6,1.25),lr(7.1,1.25),0,(default=x'30313233')
INDEX1   = 'Account',FIELD1,(TYPE=GROUP)
```

Field based on a floating trigger:

```
TRIGGER1=UL(5.75,0.71),LR(7.93,1.06),*,'Bill Summary'
TRIGGER2=UL(1.82,7.56),LR(3.40,7.85),*,'Account Number',(TYPE=FLOAT)
FIELD1=UL(1.90,7.74),LR(3.24,8.04),0,(TRIGGER=2,BASE=0,DEFAULT='N/A')
FIELD2=UL(5.79,0.13),LR(8.25,0.34),0,(TRIGGER=1,BASE=0)
INDEX1='acctnum',FIELD1,(TYPE=GROUP)
INDEX2='name',FIELD2,(TYPE=GROUP)
```

Constant field syntax

FIELD*n*='*constant*'

If all of the fields are constants, a trigger is not required. For example:

```
COORDINATES=IN
FIELD1='WASHINGTON,DC-MD-VA-WV'
FIELD2=',USA'
INDEX1='phys_loc',FIELD1,FIELD2,(TYPE=GROUP)
INDEXSTARTBY=1
```

Constant field 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 2000 characters in length. The PDF indexer does not validate the type or content of the constant. You can specify the constant value in hexadecimal.

Constant field examples

The following field parameter causes the PDF indexer to store the same text string in each INDEX1 value it creates.

```
FIELD1='000000000'
INDEX1='acct',FIELD1
```

The following field parameters cause the PDF indexer to concatenate a constant value with the index value extracted from the data. The PDF indexer concatenates the constant value specified in the FIELD1 parameter to each index value located using the FIELD2 parameter. The concatenated string value is stored in the database. In this example, the account number field in the data is 14 characters in length. However, the account number in the database is 19 characters in length. Use a constant field to concatenate a constant five character 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)
```

Using a regular expression to extract a date from a field in the form of 'July 4, 1956':

```
FIELD1=UL(0.54,0.40),LR(1.64,0.67),0,(TRIGGER=1,BASE=0,
REGEX='[A-Z][a-z]+ [0-9]+, [0-9]{4}',DEFAULT='January 1, 1970')
INDEX1='RDate',FIELD1
```

Related parameters

- "INDEX" on page 232
- "TRIGGER" on page 239

FONTLIB

Identifies the directory or directories in which fonts are stored. Specify any valid path. The PDF indexer searches for fonts in the order that the paths are listed. If a font is referenced in an input file but not embedded in the file, the PDF indexer attempts to locate the font in the directory or directories listed on the **FONTLIB** parameter.

If you purchase additional fonts and install them on the system, the additional fonts can be found at indexing time by specifying the location with the **FONTLIB** parameter.

Required?
No

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/Acrobat9/Fonts:/opt/IBM/ondemand/V10.5/fontlib
```

or

```
FONTLIB=/QIBM/ProdData/OnDemand/Adobe/Fonts
```

The PDF indexer searches the paths in the order in which they are specified. Delimit path names in UNIX and IBM i with the colon (:) character. Delimit path names in Windows with the semicolon (;) character.

A maximum of 6 paths can be specified.

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 128 index parameters. When you create index parameters, you should name the index the same as the application group database field name.

Required?
Yes

Default Value
<none>

Remember: Running the ARSPDOCI program is not supported on z/OS or Linux for System z platforms.

Syntax

INDEXn='*name*',FIELDn[,...FIELDn][,(TYPE=GROUP)]

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. You should name the index the same as the application group database field name.

You can specify the index name in hexadecimal. If you specify the value in hexadecimal, it must be specified in UTF-8.

The name used in the INDEX parameter must match the Load ID Name value provided on the Load Information tab for the application group database field name.

FIELD*nn*

The name of the field parameter or parameters that the PDF indexer uses to locate the index. You can specify a maximum of 128 field parameters. Separate the field parameter names with a comma. If the index contains a field which is based on a floating trigger, it must be the only field in the index.

When the index value is constructed, the total length of all the concatenated index values cannot exceed 2000 characters.

TYPE=GROUP
PDF indexer supports group-level indexes only. This parameter is optional.

Examples

The following index parameter causes the PDF indexer to create group-level indexes for date index values (the PDF indexer supports group-level indexes only).

When the index value changes, the PDF indexer closes the current group and begins a new group.

```
TRIGGER1=UL(5.75,0.71),LR(7.93,1.06),*,'Bill Summary'
FIELD1=UL(5.79,0.13),LR(8.25,0.34),0,(TRIGGER=1,BASE=0)
INDEX1='report_date',FIELD1
```

The following index parameters cause the PDF indexer to create group-level indexes for customer name and account number index values. The PDF indexer closes the current group and begins a new group when either the customer name or the account number index value changes.

```
TRIGGER1=UL(5.75,0.71),LR(7.93,1.06),*,'Bill Summary'
FIELD1=UL(5.79,0.13),LR(8.25,0.34),0,(TRIGGER=1,BASE=0)
```

```
FIELD2=UL(1.90,7.74),LR(3.24,8.04),0,(TRIGGER=1,BASE=0)
INDEX1='name',FIELD1
INDEX2='acct_num',FIELD2
```

The following index parameters cause the PDF indexer to create group-level indexes for customer name and balance index values. The PDF indexer closes the current group and begins a new group only when the customer name index value changes.

```
TRIGGER1=UL(5.75,0.71),LR(7.93,1.06),*,'Bill Summary'
TRIGGER2=UL(3.13,3.27),LR(5.59,4.32),*,'Total Balance',(TYPE=FLOAT)
FIELD1=UL(5.79,0.13),LR(8.25,0.34),0,(TRIGGER=1,BASE=0)
FIELD2=UL(1.90,7.74),LR(3.24,8.04),0,(TRIGGER=2,BASE=TRIGGER,DEFAULT='N/A'))
INDEX1='name',FIELD1
INDEX2='balance',FIELD2
```

Related parameters
"FIELD" on page 227

INDEXDD

Specifies the name or the full path name of the index object file. The PDF indexer writes indexing information to the index object file. If you specify the file name without a path, the PDF indexer puts the index object file in the current directory. If you do not specify the **INDEXDD** parameter, the PDF indexer writes indexing information to the file INDEX.

Required?
No

Note: When you process input files with the ARSLOAD program, the **STRMONOND** command, or the **ADDRPTOND** command, the PDF indexer ignores any value that you may specify for the **INDEXDD** parameter. 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. Running the ARSPDOCI program is not supported on z/OS .

Default Value
INDEX

Syntax

INDEXDD=*filename*

Options and values
The *filename* is a valid filename or full path name.

Remember:

1. Filenames and path names are case sensitive in UNIX environments, but not in Windows or IBM i.
2. If you specify the file name without a path, the PDF indexer writes the index object file to the current directory.

INDEXMODE

Determines whether the PDF Indexer uses metadata or internal indexes instead of triggers, fields, and indexes. If not specified, the PDF indexer uses the**TRIGGER**, **FIELD**, and **INDEX** parameters to perform the indexing.

Required?
No

Default Value
<none>

If INDEXMODE is specified along with **TRIGGER**, **FIELD**, or **INDEX** parameters, they are ignored.

Syntax

INDEXMODE=*mode*[,*n*]

Options and values

The *mode* variable can be any of the following values:

METADATA
Use metadata indexes to perform indexing.

INTERNAL
Use internal indexes. If INTERNAL is specified, *n* refers to the maximum number of indexes in a group. The default is 32.

Examples

The following parameters cause the IBM Content Manager OnDemand to extract metadata indexes and create a resource file. No other parameters are required.

```
RESTYPE=ALL
INDEXMODE=METADATA
```

The following parameters cause the IBM Content Manager OnDemand to extract internal indexes and create a resource file. No other parameters are required.

```
RESTYPE=ALL
INDEXMODE=INTERNAL,3
```

INDEXSTARTBY

Determines the page number by which the PDF indexer must locate the first group (document) within the input file.

The first group is identified when all of the group triggers and the fields based on them are found. Fields based on floating triggers are ignored when determining the INDEXSTARTBY page. For example, with the following parameters:

```
TRIGGER1=ul(4.72,1.28),lr(5.36,1.45),*,'ACCOUNT'
     TRIGGER2=ul(6.11,1.43),lr(6.79,1.59),1,'SUMMARY'
     INDEX1='Account',FIELD1,FIELD2
     FIELD1=ul(6.11,1.29).lr(6.63,1.45),2
     FIELD2=ul(6.69,1.29),lr(7.04,1.45),2
     INDEX2='Total',FIELD3
     FIELD3=ul(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 PDF indexer stops processing if it does not locate the first group by the specified page number. This parameter is optional, but the default is that the PDF indexer must locate the first group on the first page of the input file. This parameter is helpful if the input file contains header pages. For example, if the input file contains two header pages, you can specify a page number one greater than the number of header pages (INDEXSTARTBY=3) so that the PDF indexer will stop processing only if it does not locate the first group by the third page in the input data.

Important: When you use INDEXSTARTBY to skip header pages, the PDF indexer does not copy the non-indexed pages to the output file. For example, if you specify INDEXSTARTBY=3 and the indexer finds the first index on page three, then it skips pages one and two. Page three is the first page in the output file.

However, if a field based on a floating trigger is collected from a page which occurs before the first group, PDF indexer includes the page where the field was found, and the pages between where the field was found and the start of the first group as part of the first group. The field will be part of the first group in the Search Results .

Required?
No

Default Value
1

Syntax

INDEXSTARTBY=*value*

Options and values

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

INPUTDD

Specifies the name or the full path name of the PDF input file that the PDF indexer will process.

Required?
No

Note: When you process input files with the ARSLOAD program, the **STRMONOND** command, or the **ADDRPTOND** command, the PDF indexer 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.

Running the ARSPDOCI program is not supported on z/OS.

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 PDF indexer searches the current directory for the specified file.

MSGDD

Specifies the name or the full path name of the file to which the PDF indexer writes error messages. If you do not specify the **MSGDD** parameter, the PDF indexer writes messages to standard error (UNIX and IBM i) or the console (Windows).

Required?
No

Note: When you process input files with the ARSLOAD program, the **STRMONOND** command, or the **ADDRPTOND** command, the PDF indexer ignores any value that you may supply for the **MSGDD** parameter. If you process input files with ARSPDOCI, then verify the value of the **MSGDD** parameter.

The ARSPDOCI program is not supported on z/OS .

Default Value

stderr (UNIX and IBM i)

console (Windows)

Syntax

MSGDD=*name*

Options and values

The *name* is the file name or full path name of the file to which the PDF indexer writes messages. On UNIX servers, file and path names are case sensitive. If you specify the file name without a path, the PDF indexer 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 **STRMONOND** command, or the **ADDRPTOND** command, the PDF indexer ignores any value that you may supply for the **OUTPUTDD** parameter. If you process input files with ARSPDOCI, then you must specify a value for the **OUTPUTDD** parameter.

The ARSPDOCI program is not supported on z/OS .

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 PDF indexer 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 **STRMONOND** command, or the **ADDRPTOND** command, the PDF indexer ignores any value that you may supply for the **PARMDD** parameter. If you process input files with ARSPDOCI, 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 PDF indexer 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.

Tip: Because this parameter rewrites the input file, it can be used to repair minor syntax errors in the PDF.

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 PDF indexer collects resources to the resource object file. If you specify the file name without a path, the PDF indexer puts the resource object file in the current directory. Use the **RESOBJDD** parameter in conjunction with the **RESTYPE** parameter for the PDF indexer to collect resources.

Required?
No

When you process input files with the ARSLOAD program, the **STRMONOND** command, or the **ADDRPTOND** command, the PDF indexer 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.

Running the ARSPDOCI program is not supported on z/OS .

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 and Linux, but are not case-sensitive on Windows or IBM i.
2. If the PDF file does not contain resources, no RESOBJDD file is produced.

RESTYPE

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

Required?
No

Default Value
`All` (when creating a new application)

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 PDF indexer uses for temporary work space.

Required?
No

Default Value
/tmp (UNIX and IBM i)

C:\TEMP (Windows)

Syntax

TEMPDIR=*directory*

Options and values

The *directory* is a valid directory name.

TRACEDD

See "Trace facility" on page 245.

Related parameters
"FIELD" on page 227

TRIGGER

Identifies locations and string values required to uniquely identify the beginning of a group and the locations and string values of fields used to define indexes. You must define at least one trigger and can define up to 16 triggers.

Required?
Yes

Default Value
<none>

Syntax

TRIGGERn=ul(*x,y*),lr(*x,y*),*page*,'*value*'|**REGEX=**'*regular _expression*',[,(TYPE = {GROUP | FLOAT})]

Options and values

n
The trigger parameter identifier. When adding a trigger parameter, use the next available number beginning with 1 (one) to 16 (sixteen).

ul(*x,y*)
The coordinates for the upper left corner of the trigger string box. The trigger string box is the smallest rectangle that completely encloses the trigger string value (one or more words on the page). The PDF indexer must find the trigger string value inside the trigger string box. The supported range of values is 0 (zero) to 200, page width and length, in inches.

lr(*x,y*)
The coordinates for the lower right corner of the trigger string box. The trigger string box is the smallest rectangle that completely encloses the trigger string value (one or more words on the page). The PDF indexer must find the trigger string value inside the trigger string box. The supported range of values are 0 (zero) to 200, page width and length, in inches.

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

- For TRIGGER1, the *page* value must be an asterisk (*), to specify that the trigger string value can be located on any page in the input file. The PDF indexer begins searching on the first page in the input file. The PDF indexer continues searching until the trigger string value is located, the INDEXSTARTBY value is reached, or the last page of the input file is searched, whichever occurs first. If the PDF indexer reaches the INDEXSTARTBY value or the last page and the trigger string value is not found, then an error occurs and indexing stops.
- For all other group 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.
- For Float Triggers, the page value must be an asterisk (*), to specify that the trigger string value can be located on any page in the input file. The PDF indexer begins searching on the first page in the input file. If the trigger is not found, this situation is not considered an error.

'*value*'
The actual string value the PDF indexer uses to match the input data. The string value is case sensitive. The value is one or more words that can be found on a page. 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 the examples for a hexadecimal trigger example. You can specify either a value or a regular expression, but not both. If you specify the value in hexadecimal, it must be specified in UTF-8.

REGEX='*regular_expression*'
The regular expression the PDF indexer uses to match the input data. Either 'value' or REGEX can be specified, but not both. The maximum length of the regular expression is 254 characters.

TYPE

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

GROUP

Triggers that identify the beginning of a group. Group triggers, and the fields based on them, define the extent of a group.

FLOAT

Triggers that identify field data that might not occur in the same location on each page, the same page in each group, or in each group. The PDF indexer searches within the trigger string box for every occurrence of the trigger. When it is found, any fields based on it will be collected. If a field is not found, the default value defined for the field will be used. Fields based on floating triggers cannot define the extent of a group.

Remember:

1. Trigger1 must be a group trigger.
2. Fields based on floating trigger must contain a default value.
3. Fields based on floating triggers cannot be combined with any other field in an index.
4. At least one index must contain a field (or fields) based on a group trigger.

TRIGGER1 example

The following TRIGGER1 parameter causes the PDF indexer to search the specified location on every page of the input data for the specified string.

You must define TRIGGER1 and the page value for TRIGGER1 must be an asterisk.

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

Group triggers example

The following trigger parameter causes the PDF indexer to attempt to match the string value `Account Number` within the coordinates provided for the trigger string box. The trigger can be found on the same page as TRIGGER1.

```
TRIGGER2=ul(1,2.25),lr(2,2.5),0,'Account Number'
```

The following trigger parameter causes the PDF indexer to attempt to match the string value `Total` within the coordinates provided for the trigger string box. In this example, a one by four inch trigger string box is defined, because the vertical position of the trigger on the page may vary. For example, assume that the page contains account numbers and balances with a total for all of the accounts listed. There can be one or more accounts listed. The location of the total varies, depending on the number of accounts listed. The field parameter is based on the trigger so that the PDF indexer can locate the field regardless of the actual location of the trigger string value. The field is a one inch box that always begins one inch to the right of the trigger. After locating the trigger string value, the PDF indexer adds the upper left coordinates of the trigger string box to the coordinates provided for the field. The trigger can be found on the page following TRIGGER1.

```
TRIGGER2=ul(4,4),lr(5,8),1,'Total'
FIELD2=ul(1,0),lr(2,1),0,(TRIGGER=2,BASE=TRIGGER)
```

Float trigger example

The following trigger parameter causes the PDF indexer to attempt to match the string value Total Balance within the coordinates provided for the trigger string box.

The field is on the same page as the trigger.

```
TRIGGER2=UL(0.57,0.71),LR(0.89,2.40),*,'Total Balance',(TYPE=FLOAT)
FIELD2=UL(1.06,1.77),LR(3.29,2.06),0,(TRIGGER=2,BASE=0,DEFAULT='N/A')
INDEX2='Balance',FIELD2,(TYPE=GROUP)
```

Hexadecimal trigger example

The following 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)
```

This example shows how to code a trigger that represents two side-by-side UTF-8 characters in a document.

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'
```

Hexadecimal trigger example for IBM i servers

Beginning at server version 9.5.0.x, any hexadecimal triggers must be changed to be specified in ASCII instead of EBCDIC. Indexing of documents will fail at server version 9.5.0.x and later if you do not change the hexadecimal triggers to ASCII.

If the trigger is the phrase System Date, the EBCDIC hexadecimal trigger is:

```
TRIGGER1=UL(0.56,0.53),LR(1.42,0.72),*,X'E2A8A2A3859440C481A385'
```

After upgrading to server version 9.5.0.x or higher, the trigger must be changed to an ASCII hexadecimal value of:

```
TRIGGER1=UL(0.56,0.53),LR(1.42,0.72),*,X'53797374656D2044617465'
```

Regular expression example

The following trigger parameter causes the PDF indexer to attempt to match, within the coordinates provided for the trigger string box, a string that begins with an uppercase 'A', followed by 6 lowercase letters, followed by a space, followed by an uppercase 'N', followed by 5 lowercase letters. Therefore, the regular expression could match the string 'Account Number'. The trigger can be found on the same page as TRIGGER1.

```
TRIGGER2=ul(1,2.25),lr(2,2.5),0,REGEX='A[a-z]{6} N[a-z]{5}',(TYPE=GROUP)
```

PDF indexer messages

The PDF indexer creates a message list at the end of each indexing run.

A return code of 0 (zero) means that processing completed without any errors.

The PDF indexer detects a number of error conditions that can be logically grouped into several categories:

Informational

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

Error

The PDF indexer issues an error message and return code of 1 (one) and terminates processing the current input file. Most error conditions detected by the PDF indexer fall into this category. The exact method of termination may vary. For certain severe errors, the PDF indexer may fail with a segment fault. This is generally the case when some system service fails. In other cases, the PDF indexer terminates with the appropriate error messages written either to standard error or to a file. When the PDF indexer is invoked by the 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.

On IBM i platforms, the **ADDRPTOND** and **STRMONOND** commands also write error messages to the system log. The messages are also written to the log of any job running the ARSLOAD, **ADDRPTOND**, or **STRMONOND** commands.

Remember: The ARSPDOCI command is not supported on z/OS or Linux for System z platforms..

Adobe Toolkit

If the Adobe libraries fail to initialize, the PDF indexer issues an error message with a PDF library return code of 16 and stops processing the current input file.

Internal Error
The PDF indexer issues an error message and return code of 1 (one) and terminates processing the current input file.

See *IBM Content Manager OnDemand: Messages and Codes* for a list of the messages that can be generated by the PDF indexer and the explanations of the messages and actions that you can take to respond to the messages. The messages that are generated by the PDF indexer are listed in the Common Server section of the messages publication.

ARSPDOCI program

The ARSPDOCI 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.

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.

Restriction: The ARSPDUMP and ARSPDOCI programs are not supported on z/OS.

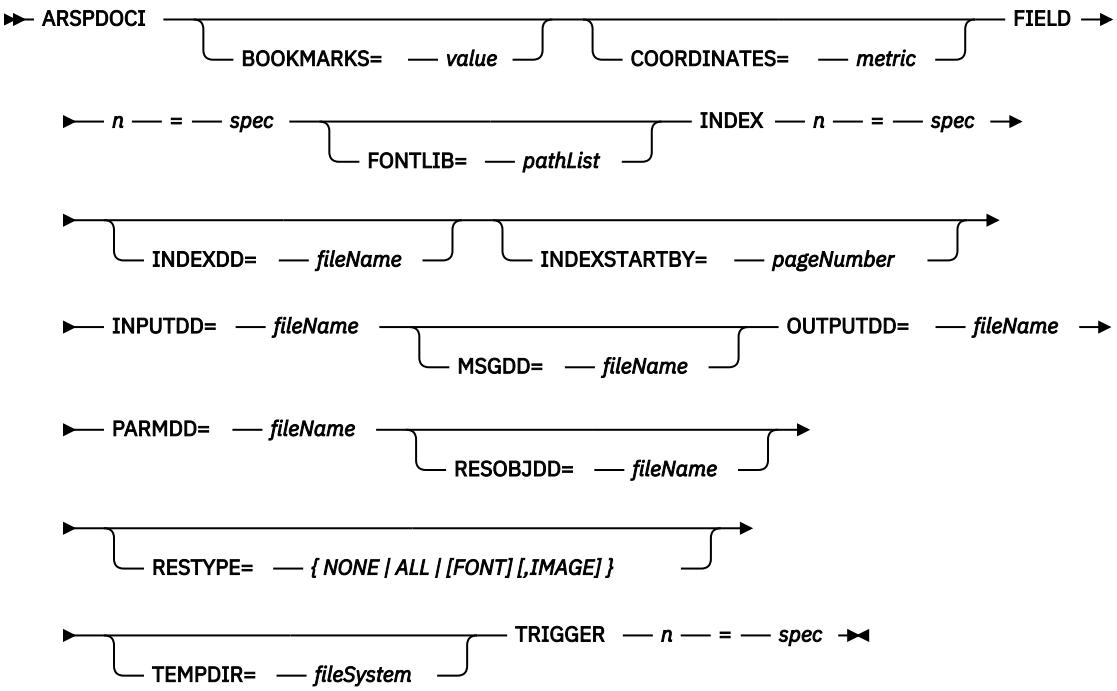
If you need to index a PDF file and you do not want to use the ARSLOAD program to process the file, then you can run the ARSPDOCI program from the command line or call it from a user-defined program.

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 or is not one of the 14 base fonts, and the PDF indexer cannot locate the font, the indexing will fail. If the customer purchases additional fonts and installs them on the system, the additional fonts can be found at indexing time by specifying the location with the **FONTLIB** parameter.

Syntax

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.



Parameters

Refer to the parameter reference 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

/opt/IBM/ondemand/V10.5/bin/arspdoci
The AIX executable program.

/opt/ibm/ondemand/V10.5/bin/arspdoci
The Linux executable program.

C:\Program Files\IBM\OnDemand\V10.5\bin\arspdoci.exe
The Windows executable program.

/usr/bin/arspdoci
The IBM i executable program.

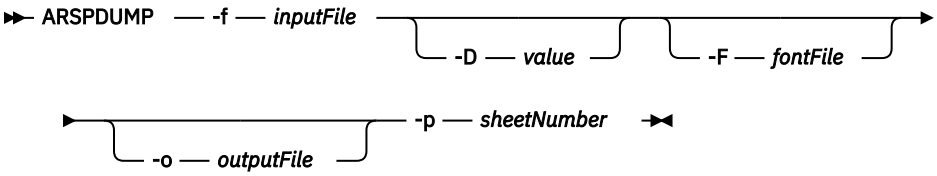
ARSPDUMP program

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, it may not be able to display the text strings in the output.

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.

Syntax



Parameters

-D value
Disable character reordering. The character reordering affects how text is extracted from PDF documents. By default, the character reordering is disabled, which is the same as setting

```
-D 1
```

. However, if the PDF page contains heavily overlapped characters, enabling the character reordering by setting

```
-D 0
```

may produce more consistent text extraction.

-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 and IBM i 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.

-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 and IBM i) 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.

Examples

- The following example shows how to invoke the ARSPDUMP program (for IBM i, within QSHELL) 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
```

- The following example shows how to invoke the ARSPDUMP program (for IBM i, within QSHELL) 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
```

Content Manager OnDemand for i users: See the *Content Manager OnDemand for i: Common Server Administration Guide* for more information about running ARSPDUMP using QSHELL.

Files

/opt/IBM/ondemand/V10.5/bin/arspdump
The AIX executable program.

/opt/ibm/ondemand/V10.5/bin/arspdump
The Linux executable program.

C:\Program Files\IBM\OnDemand\V10.5\bin\arspdump.exe
The Windows executable program.

/usr/bin/arspdump
The IBM i executable program.

Trace facility

The tracing capability for the PDF indexer provides assistance to users attempting to debug problems, such as when the system fails during the indexing and loading of PDF documents.

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

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

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

```
arspdoci parmdd=filen.parms inputdd=filen.pdf outputdd=filen.out indexdd=filen.ind
tracedd=filen.trace tracelevel=PDF=15
```

Where:

- `arspdoci` is the name of the command-line version of the PDF indexer program.
- `parmdd=` specifies the name of the input file that contains the indexing parameters.
- `inputdd=` specifies the name of the PDF input file to process.
- `outputdd=` specifies the name of the output file that contains the indexed PDF documents created by the PDF indexer.
- `indexdd=` specifies the name of the output file that contains the index information that will be loaded into the database.
- `tracedd=` specifies the name of the output file that contains the trace information.
- `tracelevel=` specifies the amount of detail to be included in the trace.

After running the PDF indexer with the trace, the output file specified by the `tracedd=` parameter will contain detailed information about the processing that took place and where the PDF indexer is failing during the process. The output file must be formatted by the arstfmt command to create a readable trace file. For example:

```
arstfmt -i filen.trace -o filen.trace.output
```

Remember: The ARSPDOCI command is not supported on z/OS or Linux for System z platforms..

Chapter 6. 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.

Content Manager 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 PDF indexer.
- For IBM i spooled files, use the 400 indexer.

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.

Content Manager 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 the ACIF, PDF, XML, or 400 indexer. The Generic indexer is not an indexing program, but refers to a parameter file format.

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 Content Manager OnDemand in the same format in which they were created. The files can be retrieved from Content Manager OnDemand and viewed with the word processor. However, because the files do not contain AFP data, line data, or PDF data, you cannot index them with the other indexers that are supported by Content Manager 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 Content Manager 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. Content Manager 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 Content Manager 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.

Related concepts

Indexing input data

Indexing parameters include information that allow the PDF indexer to identify key items in the print data stream, *tag* these items, and create *index elements* pointing to the tagged items.

Loading data

You can use the ARSLOAD program to load data into Content Manager OnDemand. If you are using Content Manager OnDemand for i, you can load data using ARSLOAD, or you can use the Start Monitor (STRMONOND) or Add Report (ADDRPTOND) command.

Loading data by using the ARSLOAD program

If the input data needs to be indexed, ARSLOAD 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, ARSLOAD can invoke the Generic indexer to process the parameter file and generate the index data. ARSLOAD 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 ARSLOAD, and an additional method on z/OS as well:

Daemon mode
The ARSLOAD program runs as a daemon (UNIX, z/OS and IBM i servers) or service (Windows servers) to periodically check a specified directory for input files to process. When running ARSLOAD 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
ARSLOAD is run from the command line (qshell on IBM i or OMVS on z/OS systems) to process a specific file. When running ARSLOAD in manual mode, specify only the *name* of the file to process. ARSLOAD 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, ARSLOAD 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.

Batch mode (z/OS only)
On z/OS , JCL can be used to start the ARSLOAD program in a UNIX System Services environment. The parameters for the ARSLOAD program are provided by using the PARM keyword on the **EXEC** statement. See the ARSLOAD command section of the *Administration Guide* for more information.

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 ARSLOAD is running in daemon mode).

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

```
MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMMSST.ARD
MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMMSST.ARD.ind
MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMMSST.ARD.out
```

The `MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMMSST.ARD` file is the dummy file that triggers a load process in daemon mode. The `MVS.JOBNAME.DATASET.FORM.YYYYDDD.HHMMSST.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.HHMMSST.ARD.out`. After successfully loading the data, the system deletes all three files.

Loading data by using the STRMONOND or ADDRPTOND commands (IBM i only)

There are two ways to run the **STRMONOND** command:

STRMONOND with TYPE(*DIR) parameter specified
The **STRMONOND** command runs as a monitor to periodically check a specified directory for input files to process. When running the **STRMONOND** command with TYPE(*DIR), the Generic indexer parameter file (`.ind`) is required to initiate a load process. The **GROUP_FILENAME:** parameter in the `.ind` file specifies the full path name of the actual input file to be processed.

STRMONOND with TYPE(*DIR2) parameter specified
The **STRMONOND** command runs as a monitor to periodically check a specified directory for input files to process. When running the **STRMONOND** command with TYPE(*DIR2), 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. This is similar to running the ARSLOAD program in daemon mode.

There is one way to run the **ADDRPTOND** command:

ADDRPTOND
The **ADDRPTOND** command is run from the command line to process a specific file. When running the **ADDRPTOND** command, you specify INPUT(*STMF) and provide the name of the `.ind` file to process in

the Stream file (STMF) parameter (omitting the .ind file extension). The **ADDRPTOND** command adds the .ind file name extension to the name that you specify. For example, if you specify STMF(po3510), where po3510 is the name of the input file, the **ADDRPTOND** command looks for and 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. This is similar to running the ARSLOAD program in manual mode.

When the data is successfully loaded, both **STRMONOND** and **ADDRPTOND** can optionally delete the input file that is specified on the **GROUP_FILENAME:** parameter if the Delete processed file (DLTSPLF) or Delete input (DLTINPUT) parameters are set to *YES. For the input file to be deleted, the input file must be located in the same directory as the file that triggered the loading of the data, the file extension must be .out, and the rest of the input file name must be the same as the .ind 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 in some cases) if the DLTSPLF or DLTINPUT parameter is set to *YES.

Example of file names for **STRMONOND** TYPE(*DIR):

```
po3510.ind
po3510.out
```

The po3510.ind file is the input file that triggers a load process for **STRMONOND** TYPE(*DIR). The po3510.ind file is the Generic indexer parameter file, and contains a **GROUP_FILENAME:** parameter that specifies the input po3510.out file to process. When the data is successfully loaded, the system deletes both files.

Example of file names for **STRMONOND** TYPE(*DIR2):

```
po3510.ARD
po3510.ARD.ind
po3510.ARD.out
```

The po3510.ARD file is the dummy file that triggers a load process for **STRMONOND** TYPE(*DIR2). The po3510.ARD.ind file is the Generic indexer parameter file, and contains a **GROUP_FILENAME:** parameter that specifies the input file to process, which is po3510.ARD.out. When the data is successfully loaded, the system deletes all three files.

If you plan to automate the data loading process on your IBM i system by using STRMONOND or ARSLOAD, one of the following must be used to identify the application group and application to load:

- the input file name
- specific parameters on the command used to load the data
- a monitor user exit program

The .ind file name extension (for STRMONOND *DIR processing) or the .ARD file name extension (for STRMONOND *DIR2 or ARSLOAD daemon processing) is required to initiate a load process. The case (uppercase or lowercase) of the extension (.ARD or .ind) is ignored. Application group and application names are case sensitive. Application group and application names might include special characters such as the blank character when using ADDRPTOND or ARSLOAD with a specific application group and application name provided. If a blank or other special character is included in the application group name or application name when used in this manner, the full name must be enclosed in single quotes. Mixed-case or lowercase names must also be enclosed in single quotes. However, when using the STRMONOND command or the MVS naming convention on the ARSLOAD API (with the -A and -G parameters), archiving files that have spaces in the file name, application group name, or application name is not supported.

If you plan to automate the data loading process for PDF files, you might choose to use the STRMONOND command with monitor type *DIR. If you choose this approach, it is important to carefully consider which directories will receive the .ind and .pdf files to be loaded. By design, the Content Manager OnDemand directory monitor (with monitor type *DIR) processes files with a .IND or .PDF extension. At the time that the STRMONOND monitor selects a file to process, Content Manager OnDemand does not know which indexer (PDF indexer or Generic indexer) will be used to process the file. It is only after the application group and application are identified from the file name, and the indexer type is determined based on the application definition, that Content Manager OnDemand knows which indexer to use.

Relying on the arrival sequence of the files into the monitored directory does not ensure that they will be processed correctly. If the input .pdf file is placed in the same directory as the index (.ind) file, and the input .pdf file is found first or the .ind file does not yet exist in the directory, one of two things might happen:

- The monitor will not find an application group name and application name to use, based on what was specified at the time the monitor was started, or
- The monitor will find an application group name and application name to use, but because the Generic indexer is specified as the indexer type in the application, it will not be able to process the file because the Generic indexer is expecting a .ind file to point to the associated .pdf file to process.

If either of these situations occur, processing of the .pdf file fails and it is renamed to include a .ERR extension. Content Manager OnDemand then picks up the next file to process which might be the .IND file that points to the .PDF file that Content Manager OnDemand just attempted to process. Content Manager OnDemand is unable to find the data file specified in the .IND file because the .PDF file has been renamed with the .ERR extension and so processing of the .IND file fails and that file is also renamed to include the .ERR extension.

In this scenario, if the input file to be processed must have a .PDF extension, the PDF file should be placed in a directory other than the directory being monitored, and it should arrive before the .IND file is placed in the monitored directory. The .IND file should reflect the correct path to the .PDF file for processing (which will be a different directory than the directory in which the .IND file is placed). Alternatively, the .PDF file could be renamed to have a file extension of .OUT (i.e., something other than .PDF). In this case, Content Manager OnDemand would skip over the .OUT file when looking for files to process and pick up the .IND file and successfully index and load the data.

If you plan to automate the data loading process for PDF files by using the STRMONOND command with monitor type *DIR2, there are no special considerations. The .pdf file is processed like any other input file for a *DIR2 monitor type.

See the *IBM Content Manager OnDemand for i: Common Server Administration Guide* for more information about using the STRMONOND and ADDRPTOND commands and the ARSLOAD API to load data.

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 20. 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`

Table 20. 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 2/End Document 2	6505 / 6267	6223 / 6267
Begin Document 3/End Document 3	12772 / 6588	12490 / 6588
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.

Generic indexer 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.

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:

- The parameter names in the parameter file are case sensitive and must appear in uppercase. For example, `GROUP_FIELD_NAME:account` is valid, while `group_field_name:account` is not.
- 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_JP97` or some other locale that correctly identifies uppercase and lowercase 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 uppercase and lowercase characters in code page 954. On Windows, set the system locale in the Region and Language dialogue.

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 Content Manager OnDemand. You specify the name of the application group to the ARSLOAD program. Content Manager OnDemand supports up to 128 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. Content Manager 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 Content Manager OnDemand. You specify the name of the application group to the ARSLOAD program. Content Manager OnDemand supports up to 128 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.

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

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_FIELD_NAME: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).

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
```

Examples

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. 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.

Examples

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
```

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
```

Chapter 7. XML indexer

IBM Content Manager OnDemand provides the XML indexer to allow you to index and archive your XML documents.

Because XML data can contain any kind of information using an endless number of tags, you must indicate to the XML indexer what documents within your data you would like stored along with the index values for each document. This is accomplished by transforming your XML input into an intermediate format. Tools like XSLT and XQuery can be used to get your data ready for indexing.

XSLT

XSLT (Extensible Stylesheet Language Transformations) is a language for transforming XML documents into other XML documents, or even plain text. During the transformation, the original document is not changed; rather, a new document is created based on the content of an existing one. The basic processing paradigm is pattern matching. The XSLT style sheet defines what patterns to process and how to process them for output. While there are many processor implementations of XSLT, Saxon and Xalan are two of the more popular open source versions.

XQuery

XQuery is a query and functional programming language. XQuery can be used to query and transform your XML data into the format defined by IBM Content Manager OnDemand for XML indexing.

The XML indexer will validate the structure and content of your input file using the XML indexer schema file named `odxmlidx.xsd`.

Important: After preparing your XML data for indexing, your input file must follow the constraints and structure defined in the schema file. Otherwise, indexing errors might occur.

.xsd schema file

The following file listing shows the `odxmlidx.xsd` schema file that is included with Content Manager OnDemand. You transform your XML data to the format in this schema file so the XML indexer can ingest it.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="odidx">
     <xs:complexType>
       <xs:choice>
        <xs:element name="oddoc" minOccurs="1" maxOccurs="unbounded">
           <xs:complexType>
             <xs:sequence>
               <xs:choice>

                 <xs:element name="oddataref">
                   <xs:complexType>
                     <xs:attribute name="file"
                                   type="xs:string"
                                   use="required"/>
                     <xs:attribute name="offset"
                                   type="xs:nonNegativeInteger"
                                   use="required"/>
                     <xs:attribute name="length"
                                   type="xs:nonNegativeInteger"
                                   use="required"/>
                   </xs:complexType>
                 </xs:element>

                 <xs:element name="odxmldata">
                   <xs:complexType>
                     <xs:sequence>
                       <xs:any minOccurs="0" processContents="skip"/>
                     </xs:sequence>
                   </xs:complexType>
                 </xs:element>
```

```
                </xs:choice>

                <xs:element name="odindex"
                            minOccurs="1"
                            maxOccurs="128">
                  <xs:complexType>
                    <xs:attribute name="field"
                                  type="xs:string"
                                  use="required"/>
                    <xs:attribute name="value"
                                  type="xs:string"
                                  use="required"/>
                  </xs:complexType>
                </xs:element>

              </xs:sequence>
            </xs:complexType>

            <xs:unique name="uniqueIndexField">
              <xs:selector xpath="./odindex"/>
              <xs:field xpath="@field"/>
            </xs:unique>
          </xs:element>
        </xs:choice>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

Resources

While XML files are readable, there are several technologies which make the viewing of XML data more user friendly. Cascading Style Sheets (CSS) and Extensible Stylesheet Languages (XSL) are two such technologies.

With both CSS and XSL, files containing the layout or formatting that is to be applied to the XML are specified. Multiple CSS or XSL files can be defined for each XML document. These files are considered resources within Content Manager OnDemand and can be archived as resources. These resources are specified in your XML data using the `<?xml-stylesheet>` processing instruction. The location of the resource file is specified using the `href` attribute of the xml-stylesheet processing instruction. In order for your resources to be collected and archived, they must be of the type `file:`. Only top level `file:` type style sheets will be collected and archived as a resource for the documents contained in the input file.

Most style sheets will contain references to other files (for example: an image for a corporate logo) and these embedded references will not be archived. To accommodate the archiving of these referenced files and any other supporting files, the XML indexer will allow you to specify the resource file to be used for the load. For XML documents, this file must be a zip file archive.

Invocation

The XML indexer is a single pass process which means that no intermediate files are produced.

To run the XML indexer, run the **ARSLOAD** program with all of the default options and specify `xml file` for the input source. The `-X` option can be used to specify an indexer other than the one specified by the application.

Notices

This information was developed for products and services that are offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide 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 Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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.

The client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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-IBMproducts. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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 actual people or business enterprises 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.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).
Portions of this code are derived from IBM Corp. Sample Programs.
© Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at www.ibm.com/privacy and IBM’s Online Privacy Statement at www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM

Software Products and Software-as-a-Service Privacy Statement" at www.ibm.com/software/info/product-privacy.

Index

Numerics

390 indexer
- about 145
- ANYEXIT parameter 147
- Anystore Batch Capture Exit 147
- application indexer 173
- Content Manager OnDemand application 173
- DD:ddname parameter 173
- index exit parameter 165
- indexer information 173
- INDXEXIT parameter 165
- INPEXIT parameter 167
- INPEXITNEW parameter 169
- input exit parameter 167, 169
- introduction 145
- line data 145
- overview 145
- parameters 147
- specifying parameters for the ARSLOAD program 173
- TIFF images 145
- using 145, 172

390 indexer example 173
400 indexer
- about 175
- introduction 175
- overview 175
- parameters 175
- TRIGGER parameter 190
- triggers 190
- using 175

A

accessibility xi
accessing reports 107
account number field
- defining 110

account number index
- defining 111

ACIF
- ACIF input record exit 75
- AFP resources 6, 21, 30, 31, 33, 49–51, 54, 56–58, 60, 68
- apka2e input record exit 75
- BREAKYES parameter 13
- CC parameter 14
- CCTYPE parameter 14
- CHARS parameter 15
- CONVERT parameter 16
- CONVERT requirement 16
- CPGID parameter 17
- DCFPAGENAMES parameter 17
- default index value 22
- description 5
- EBCDIC data 8
- examples 97

ACIF *(continued)*
- exit 75
- exit, index 76
- exit, input file 72
- exit, output record 78
- exit, resource retrieval 80
- exit, user programming 72
- exits 85
- extended options 18
- EXTENSIONS parameter 18
- FDEFLIB parameter 21
- FIELD parameter 22
- fields 22, 100, 110, 120
- FILEFORMAT (z/OS platforms) parameter 29
- FILEFORMAT (Multiplatform) parameter 28
- FONTLIB parameter 30
- FORMDEF parameter 31
- FORMFEED parameter 33
- group indexes 34, 35
- group-level indexes 7
- GROUPMAXPAGES parameter 34
- GROUPNAME parameter 34
- IMAGEOUT parameter 35
- index exits 76
- INDEX parameter 35
- index, exit 72
- INDEXDD parameter 38
- Indexer Information page 99
- indexes 35, 101, 111, 123
- indexing 72
- INDEXOBJ parameter 39
- INDEXOBJ requirement 39
- INDEXSTARTBY parameter 40
- INDXEXIT parameter (z/OS platforms) 41
- INDXEXIT parameter (Multiplatform) 41
- INPEXIT parameter (z/OS platforms) 42
- INPEXIT parameter (Multiplatform) 42
- input exits 72
- input file, exit 72
- input record exits 75
- input, user exit 71
- INPUTDD parameter (z/OS platforms) 43
- INPUTDD parameter (Multiplatform) 42
- INSERTIMM parameter 43
- introduction 3
- Invoke Medium Map structured field 139
- invoking program to index input file 133
- JCL statement defined 131
- large object support 39
- line data 5
- LINECNT parameter 44
- load information 103
- mask 69
- MCF2REF parameter 45
- message file 131
- messages 71
- MSGDD parameter (z/OS platforms) 46

ACIF *(continued)*
MSGDD parameter (Multiplatform) 46
NEWPAGE parameter 47
non-zero return codes 84
OUTCCSID parameter 47
OUTEXIT parameter (z/OS platforms) 48
OUTEXIT parameter (Multiplatform) 48
output file format 96
output record exits 78
OUTPUTDD parameter (z/OS platforms) 49
OUTPUTDD parameter (Multiplatform) 48
OVLYLIB parameter (z/OS platforms) 50
OVLYLIB parameter (Multiplatforms) 49
page indexes 16, 35, 39
page-level indexes 3
page-level indexes not stored in database 7
PAGEDEF parameter 51
pagerange indexes 35
parameter file 131
parameter reference 11
parameters syntax 132
parameters to use 129
parameters, z/OS 132
PARMDD parameter (z/OS platforms) 53
PARMDD parameter (Multiplatform) 53
PDEFLIB parameter 54
print file attributes 84
PRMODE parameter 55
program requirements 131
PSEGLIB parameter 56
reference 3
requirements 3, 131
RESEXIT parameter 57
RESFILE parameter 57
RESLIB parameter 58
RESOBJDD parameter (z/OS platforms) 59
RESOBJDD parameter (Multiplatform) 59
resource provided with ACIF 80
resource retrieval 80
resources 21, 30, 31, 33, 49–51, 54, 56–58, 60, 68
RESTYPE parameter 60
return code, non-zero 84
specifying triggers 8
syntax rules, z/OS 132
TLE structured fields 127
TRACE parameter 62
TRACEDD parameter (z/OS platforms) 63
TRACEDD parameter (Multiplatform) 63
transaction fields 100
TRC parameter 63
TRIGGER parameter 64
triggers 64, 109, 118
UNIQUEBNGS parameter 67
user exit input 71
user exit provided with ACIF 72
user exit search order 83
user exit, print file attributes 84
user programming exit 72
USERLIB parameter 68
USERMASK parameter 69
USERPATH parameter 70
using 131
z/OS JCL statement 131
z/OS requirements 131
ACIF indexer
regular expressions 141, 195
ACIF parameters
INDEXSTARTBY 158
adding an application
last step 130
ADDRPTOND command 247
Adobe font requirements
PDF indexer 223
Adobe PDF documents 209
AFP
ARSACIF
programs 140
concatenation 140
converting line data to 5
example of converting line data to 104, 114
example of indexing 127
fonts 30, 60
form definitions 21, 31, 33
generic indexer, processing with 250
IMM structured fields 43
indexing with the generic indexer 250
MCF2 structured fields 45
overlays 49, 50
page definitions 51, 54
page segments 56
parameters to use 129
processing with the generic indexer 250
programs
ARSACIF 140
resources 21, 30, 31, 33, 49–51, 54, 56–58, 60, 68
Set Coded Font Local structured fields 55
SOSI 55
user-defined resources 68
AFP Application Programming Interface
Tag Logical Element 87
AFP resources
collecting 6
description 6
AFPDS
converting to AFP data 5
placing TLEs in named groups 135
printing 136
transferring 138
AFPDS (AFP data stream)
description 5
object support 5
resource support 5
ANSI carriage controls 137
ANYEXIT parameter
390 indexer 147
Anystore Batch Capture Exit
390 indexer 147
applications
adding 104, 114
defining 98, 117
archiving, ACIF
indexing considerations 139
ARSLOAD program
specifying parameters for the 390 indexer 173
ARSPDOCI
error messages 241
FIELD parameter 227
FONTLIB parameter 231

ARSPDOCI *(continued)*
INDEX parameter 232
INDEXDD parameter 233
INPUTDD parameter 235
messages 241
MSGDD parameter 235
OUTPUTDD parameter 236
reference 226, 242
TEMPDIR parameter 238
ARSPDUMP program
reference 243
attribute names
mapping to database field names 130
attributes
print file 84

B

BCOCA value 60
Begin Document Index structured field
defined 91
Begin Document structured field
defined 94
Begin Named Group structured field
defined 95
Begin Page structured field
defined 95
Begin Resource Group structured field
described 88
Begin Resource structured field
defined 89
bill date field
defining 111
bookmarks
PDF indexer 224
BOOKMARKS 226
BOX
EXTENSIONS parameter 18
BPF 96
BREAKYES parameter
flags and values 13
syntax 13
valid values 13

C

carriage controls 14, 44
carriage-control characters
indexing considerations 139
CC parameter
400 indexer 177
flags and values 14
related parameters 14
syntax 14
valid values 14
CCTYPE parameter
400 indexer 177
flags and values 14
syntax 14
valid values 14
CELLED
EXTENSIONS parameter 18
CHARS parameter
CHARS parameter *(continued)*
flags and values 15
specifying value 15
syntax 15
CMRALL value 60
CMS commands
invoking ACIF program to index input file 133
COBOL 85
code page
data indexing 17
generic indexer 251
PDF indexer 225
report file 17
CODEPAGE: parameter 251
collecting AFP resources
description 5
commands
ARSPDOCI 242
ARSPDUMP 243
COMMENT: parameter 252
comments
in parameter file 132
Composed Text Control (CTC) structured field
obsolete 96
concatenation
z/OS files 134
constant field 227
Constant FIELD syntax
400 indexer 180
Content Manager OnDemand application
indexer information 173
specifying 390 indexer 173
conversion 16
CONVERT parameter
400 indexer 178
flags and values 16, 43
syntax 16
valid values 16
converting print data streams
description 5
coordinate system 214
coordinates
FIELD parameter
mask option 227
fields
mask option 227
indexing
field mask 227
mask option 227
mask
FIELD parameter option 227
on FIELD parameter for PDF indexer 227
PDF indexer
field mask 227
mask option 227
COORDINATES parameter
flags and values 226, 233
INDEXMODE
syntax 233
CPGID parameter
390 indexer 151
400 indexer 178
flags and values 17
syntax 17

CPGID parameter *(continued)*
valid values 17
creating index parameters
instructions 129
customer name field
defining 110
customer name index
defining 112

D

data
format 28, 29
database field
mapping to attribute names 130
date field 205
DBCS fonts 223, 225
DCB requirements
message file, z/OS 131
output file, z/OS 131
DCFPAGENAMES parameter
flags and values 17
syntax 17
valid values 17
DD:ddname parameter 173
default index value
FIELD parameter option 22, 227
defining an application
introduction to 129
defining fields 100
defining indexes 111, 123
defining triggers 109
diagnostic trace information 62
DISABLECHARREORDERING parameter
flags and values 227
DJDE record 172
DJDECNT parameter 151
DJDECOL parameter 152
DJDETRIG parameter 152
DOC 158
DOCTYPE parameter
400 indexer 178
document
DD statement for, z/OS 131
generic indexer parameter 253, 255
output format 93

E

EBCDIC
parameter file for input data 10
EBCDIC data
CCTYPE parameter 14
CPGID parameter 17
example of 8
indexing 8
specifying 8
TRIGGER parameter 64, 190
USERMASK parameter 69
edition notice 2
EMPTYOK
EXTENSIONS parameter 18
End Document Index structured field
defined 93
End Document Index structured field *(continued)*
defined 93
End Document structured field
defined 95
End Named Group structured field
defined 95
End Page structured field
defined 95
End Resource Group structured field
defined 89
End Resource structured field
defined 89
EPF 96
error messages
ARSPDOCI program 241
PDF indexer 241
examples
AFP data, indexing 104, 114, 127
AFP document output formats 93
EBCDIC input data, parameter file for 10
generic indexer 256
indexing 97, 104, 114
invoking ACIF program to index input file 133
JCL and ACIF processing parameters 133
line data, converting to AFP 104, 114
line data, indexing 97, 104, 114
print file attributes 84
z/OS JCL to invoke ACIF 131
exits
index 76
input 72
non-zero return codes 84
output 78
print file attributes provided 84
resource, provided with ACIF 80
user exit search order 83
extended options 18
EXTENSIONS parameter
flags and values 18
related paramters 18
RESORDER value 60, 141
syntax 18

F

FDEF value 60
FDEFLIB parameter
flags and values 21
related parameters 21
syntax 21
valid values 21
FIELD parameter
390 indexer 152
400 indexer 179
constant field 227
default index value 22, 227
flags and values 22, 227
how Content Manager OnDemand uses 211
trigger field 227
fields
390 indexer 152
ACIF parameter 22
constant field 227
default index value 22, 227

fields *(continued)*
defining 100, 110, 120
displaying 102, 112
generic indexer parameter 252, 253
PDF indexer parameter 227
transaction fields 100
trigger field 227
file
message, ACIF 131
parameter, ACIF 131
FILEFORMAT parameter
390 indexer 155
flags and values
Multiplatform 28
z/OS platforms 29
syntax 28, 29
valid values 28, 29
files
390 indexer 155
format 28, 29
PDF indexer 224
flags and values
REMOVERES 237
RESOBJDD 237
RESTYPE 238
floating triggers 34, 64, 190
FONT value 60
FONTLIB parameter
flags and values 30, 231
related parameters 30
syntax 30
valid values 30
fonts
CHARS parameter 15
converting 45
DBCS 223, 225
directory 30
library 30
location 30
Map Coded Font Format 2 structured fields 45
MCF2 structured fields 45
NLS 55, 223, 225
PDF indexer 223, 224, 231
resources 60
Set Coded Font Local structured fields 55
SOSI 55
specifying 15
TRCs 63
form definitions 21, 31, 33
FORMDEF
required by ACIF to process AFP 130
FORMDEF parameter
flags and values 31
related parameters 31
syntax 31
valid values 31
FORMFEED parameter
flags and values 33
related parameters 33
syntax 33
valid values 33
FRACLINE
EXTENSIONS parameter 18

G

general page
description 98
General page
application groups 107, 117
description 107, 117
introduction 129
generic indexer
about 247, 250
AFP data, processing 250
application group field names 252
code page 251
CODEPAGE: parameter 251
COMMENT: parameter 252
document 253, 255
examples 256
field names 252
field values 253
group indexes, defining 252, 253
GROUP_FIELD_NAME: parameter 252
GROUP_FIELD_VALUE: parameter 253
GROUP_FILENAME: parameter 253
GROUP_LENGTH: parameter 255
GROUP_OFFSET: parameter 255
input file 253, 255
introduction 247
national language support (NLS) 251
NLS 251
overview 247
parameter file 251, 256
using 247
GOCA value 60
graphical indexer
PDF input files 211
group indexes
defining 35, 232, 252
defining for generic indexer 253
pages in a group 34
GROUP_FIELD_NAME: parameter 252
GROUP_FIELD_VALUE: parameter 253
GROUP_FILENAME: parameter 253
GROUP_LENGTH: parameter 255
GROUP_OFFSET: parameter 255
group-level indexes
about 7
TLE structured fields 127
GROUPMAXPAGES parameter
390 indexer 155
flags and values 34
related parameters 34
syntax 34
valid values 34
GROUPNAME parameter
flags and values 34
syntax diagram 34
grouprange index 35
Grouprange index 35

H

header pages
skipping 40, 234

I

IMAGEOUT parameter
  400 indexer 185
  flags and values 35
IMM structured fields 43
income index
  defining 124
Index Element structured field
  considerations 139
  defined 92
  group-level 91
  index object file 139
index exit 76
Index Exit
  description 165
index exit parameter
  390 indexer 165
index information
  how Content Manager OnDemand uses 211
index object file
  DD statement for, z/OS 131
INDEX parameter
  390 indexer 156
  400 indexer 185
  flags and values 35, 232
  how Content Manager OnDemand uses 211
  JCL statement, z/OS 131
  z/OS, JCL statement 131
index parameters
  Grouprange 35
index user exit 41
INDEXDD parameter
  flags and values 38, 233
indexer
  400 175
  overview 1
indexer information
  specifying for 390 indexer 173
Indexer Information page
  description 117
  overview 129
indexer parameters
  using break=yes versus break=no 193
Indexer Parameters window
  introduction 130
indexer properties
  defining 125
  setting 113
indexes
  ACIF parameter 35
  defining 101, 111, 123
  displaying 102, 112
  generic indexer parameter 253
  group index 35
  grouprange index 35
  page index 35
  pagerange index 35
  PDF indexer parameter 232
indexing
  390 indexer 145
  400 indexer 175
  Adobe PDF documents 209
  constant field 227
indexing *(continued)*
  CONVERT requirement 16
  default index value 22, 227
  EBCDIC data 8, 14, 17, 64, 69, 190
  effect on document 93
  fields 22, 100, 110, 120
  fields for PDF indexer 227
  file format 28, 29
  floating triggers 64, 190
  generic indexer 247
  graphical indexer 211
  group indexes 35, 232
  group-level indexes 7
  groups 34
  header pages 40, 234
  helpful hints 139
  index exit 72
  indexes 35, 111, 123, 232
  INDEXOBJ requirement 39
  large object support 39
  line data 145
  line separator 28, 29
  mask 69
  new line character 28, 29
  page indexes 16, 35, 39
  page-level indexes 3
  page-level indexes not stored in database 7
  pagerange indexes 35
  parameters 214
  PDF indexer 209
  recordrange triggers 64, 190
  skipping header pages 40, 234
  TIFF images 145
  TLE structured fields 127
  transaction fields 100
  trigger field 227
  triggers 64, 109, 118, 190, 239
  XML indexer 259, 260
indexing parameter
  creating 8
indexing parameters
  specifying triggers 8
INDEXMODE
  options and values 233
INDEXOBJ
  example 130
INDEXOBJ parameter
  400 indexer 187
  flags and values 39
INDEXSTARTBY parameter
  400 indexer 187
  flags and values 40, 234
INDEXSTYLE parameter
  400 indexer 188
  AFP 163
  DOC 158
  NODX 162
  PAGE 159
  PDOC 160
INDXEXIT parameter
  390 indexer 165
  flags and values
    Multiplatform 41
    z/OS platforms 41

inline resources
output files 141
processing 18, 60, 141
structured fields
Begin Document (BDT) 141
INLINE value 60
INLONLY value 60
INPCCSID
flags and values 41
INPEXIT parameter
390 indexer 167
flags and values
Multiplatform 42
z/OS platforms 42
INPEXITNEW parameter
390 indexer 169
input
z/OS 131
input exit parameter
390 indexer 167, 169
input file
exit 72
generic indexer parameter 253, 255
input record exit
apka2e 75
input user exit 42
INPUTDD parameter
flags and values
Multiplatform 42
z/OS platforms 43
Invoke Medium Map
structured field 139
IOCA value 60

J

JCL
ACIF JCL statement defined 131
concatenating ACIF files, z/OS 134
concatenation example, z/OS 134
example, z/OS 133
for ACIF job, z/OS 133
for ACIF z/OS jobs 131
for ARSLOAD job, 390 indexer 173
for concatenating z/OS files 134
invoking ACIF program to index input file 133
OUTPUT JCL statement defined 131
PRINTOUT JCL statement defined 131
statement defined, ACIF JCL 131
statement defined, OUTPUT JCL 131
statement defined, PRINTOUT JCL 131
z/OS example 133, 134
job run date 205

K

key concepts
convert 107
group index 107
group trigger 107
index break 107
resources 107

L

large object support 39
large objects
390 indexer 173
limitations
PDF indexer 224
line data
AFP
converting to 104, 114
converting to AFP 5, 104, 114
description 6
example of indexing 97, 104, 114
groups 34
indexing 97, 104, 114
indexing with the 390 indexer 145
pages in a group 34
line separator 28, 29
LINECNT parameter
flags and values 44
LINEOFFSET parameter 170
links
PDF indexer 224
literal values
ASCII
parameter file for input data 10
determining how expressed 10
examples
ASCII input data, parameter file for 10
parameter file
ASCII input data, example of 10
load information
description 103
Load Information page
description 114
Loading data 247

M

machine carriage controls 137
Map Coded Font Format 1 structured field
converted 96
Map Coded Font Format 2 structured field
archival, document integrity 96
converting 45
including fonts 60
mask 69
Mask FIELD syntax
400 indexer 183
maximum pages in a group 34
MCC2ANSI parameter 171
MCF2 structured fields 45, 60
MCF2REF parameter
flags and values 45
message file
DD statement for, z/OS 131
messages
ACIF 71
ARSPDOCI program 241
PDF indexer 241
metadata
concepts 217, 218
indexing concepts 217, 218
mixed-mode data

mixed-mode data *(continued)*
description 6
MO:DCA-P data stream
ACIF changes to structured fields 5
defined 5
MSGDD parameter
flags and values
Multiplatform 46
z/OS platforms 46

N

naming input files
PDF indexer 224
national language support (NLS)
ACIF 17, 55
generic indexer 251
PDF indexer 223, 225
new line separator 28, 29
NEWPAGE parameter
flags and values 47
NLS
ACIF 17, 55
generic indexer 251
PDF indexer 223, 225
NODX 162
non-zero return codes 84
notices 2

O

OBJCON value 60
odxmlidx.xsd file 259
OUTCCSID 47
OUTEXIT parameter
flags and values
Multiplatform 48
z/OS platforms 48
output file
format 93
OUTPUT JCL statement
defined, z/OS 131
z/OS 131
output record exit 78
output user exit 48
OUTPUTDD parameter
flags and values
Multiplatform 48
z/OS platforms 49
overlays 49, 50
OVLY value 60
OVLYLIB parameter
flags and values
Multiplatforms 49
z/OS platforms 50

P

PAGE 159
page definition
resource file 96
page definitions 51, 54
page indexes
page indexes *(continued)*
about 94
CONVERT requirement 16
defining 35
INDEXOBJ requirement 39
large object support 39
page segments 56
page-level IELs 91
page-level indexes
about 3
not stored in database 7
TLE structured fields 127
PAGEDEF parameter
flags and values 51
pagerange index 35
parameter file
ARSPDOCI program 226, 242
comments 132
DD statement for, z/OS 131
EBCDIC input data, example of 10
generic indexer 256
PDF indexer 214, 226, 242
syntax rules, z/OS 132
values spanning multiple records 132
parameter values
spanning multiple records 132
parameters
390 indexer 147
ANYEXIT parameter for 390 indexer 147
Anystore Batch Capture Exit for 390 indexer 147
ARSPDOCI program 226, 242
ARSPDUMP program 243
BREAKYES 13
BREAKYES parameter 151
CC 14
CCTYPE 14
CHARS 15
CODEPAGE: 251
COMMENT: 252
CONVERT 16
COORDINATES 226
CPGID 17
DCFPAGENAMES 17
DISABLECHARREORDERING 227
DJDECNT parameter 151, 172
DJDECOL parameter 152, 172
DJDETRIG parameter 152, 172
examples 227
EXTENSIONS 18
FDEFLIB 21
FIELD 22, 147, 227
FILEFORMAT 147
FILEFORMAT (z/OS platforms) 29
FILEFORMAT (Multiplatform) 28
FONTLIB 30, 231
FORMDEF 31
FORMFEED 33
generic indexer 251
GROUP_FIELD_NAME: 252
GROUP_FIELD_VALUE: 253
GROUP_FILENAME: 253
GROUP_LENGTH: 255
GROUP_OFFSET: 255
GROUPMAXPAGES 34, 147

parameters *(continued)*
GROUPNAME 34
IMAGEOUT 35, 140
INDEX 35, 147, 232
index exit parameter for 390 indexer 165
INDEXDD 38, 233
INDEXMODE 233
INDEXOBJ 39
INDEXSTARTBY 40, 234
INDXEXIT (z/OS platforms) 41
INDXEXIT (Multiplatform) 41
INDXEXIT parameter for 390 indexer 165
INPCCSID 41
INPEXIT (z/OS platforms) 42
INPEXIT (Multiplatform) 42
INPEXIT parameter for 390 indexer 167
INPEXITNEW parameter for 390 indexer 169
input exit parameter for 390 indexer 167, 169
INPUTDD 235
INPUTDD (z/OS platforms) 43
INPUTDD (Multiplatform) 42
INSERTIMM 43
IOCA images 140
LINECNT 44
LINEOFFSET parameter 170
MCC2ANSI parameter 171
MCF2REF 45
MSGDD 235
MSGDD (z/OS platforms) 46
MSGDD (Multiplatform) 46
NEWPAGE 47
OUTCCSID 47
OUTEXIT (z/OS platforms) 48
OUTEXIT (Multiplatform) 48
OUTPUTDD 236
OUTPUTDD (z/OS platforms) 49
OUTPUTDD (Multiplatform) 48
OVLYLIB (z/OS platforms) 50
OVLYLIB (Multiplatforms) 49
PAGEDEF 51
PARMDD 236
PARMDD (z/OS platforms) 53
PARMDD (Multiplatform) 53
PDEFLIB 54
PDF indexer 214, 226, 242
PRMODE 55
PSEGLIB 56
RESEXIT 57
RESFILE 57
RESLIB 58
RESOBJDD (z/OS platforms) 59
RESOBJDD (Multiplatform) 59
RESTYPE 60
TEMPDIR 238
TRACE 62
TRACEDD (z/OS platforms) 63
TRACEDD (Multiplatform) 63
TRC 63
TRIGGER 64, 147, 190, 239
UNIQUEBNGS 67
USERLIB 68
USERMASK 69
USERPATH 70
z/OS 132
PARMDD parameter
flags and values
Multiplatform 53
z/OS platforms 53
PASSPF
EXTENSIONS parameter 18
PDEFLIB parameter
flags and values 54
PDF indexer
about 209
Adobe font requirements 223
ARSPDOCI reference 242
ARSPDUMP reference 243
bookmarks 224
code page 225
constant field 227
coordinate system 214
DBCS fonts 223, 225
default index value 227
error messages 241
fields 227
file naming conventions 224
font requirements 223
fonts 224, 231
graphical indexer 211
group indexes 232
indexes 232
indexing data 214
introduction 209
limitations 224
links 224
messages 241
naming input files 224
national language support (NLS) 223, 225
NLS 223, 225
overview 209
parameter file 214
parameter reference 226, 242
printing 224
regular expressions 220
resource collection 223
restrictions 224
support for DBCS fonts 224
system limitations 224
tags 214
transferring input files to 224
trigger field 227
triggers 239
using 209
x, y coordinate system 214
PDF resource collection 223
PDOC 160
Portable Document Format (PDF) 209
PostScript file
how processed by PDF indexer 224
relation to PDF file 209
PostScript Passthrough markers
PDF indexer limitations 224
PRCOLOR
EXTENSIONS parameter 18
print file attributes
user exits 84
printing
PDF indexer 224

PRINTOUT JCL statement
  defined 131
PRMODE parameter
  flags and values 55
PSEG value 60
PSEGLIB parameter
  flags and values 56
PTOCA value 60
publication
  audience xi
publication information xi

R

recordrange triggers 64, 190
REGION size for ACIF 131
regular expressions
  PDF indexer 220
REMOVERES 237
report wizard 175
reports
  accessing 8, 98, 107, 115
  example of 107, 115
  example of accessing 128
  examples
    reports 98
  format 28, 29
  opening 99, 108, 118
  transmitting data 8
requirements
  Adobe font requirements 223
  fonts 223
RESEXIT parameter
  flags and values 57
RESLIB parameter
  flags and values 58
RESOBJDD 237
RESOBJDD parameter
  flags and values
    Multiplatform 59
    z/OS platforms 59
RESOBJDD statement
  z/OS 131
RESORDER
  EXTENSIONS parameter 18
RESORDER value 60
resource exit
  provided with ACIF 80
resource file
  DD statement for, z/OS 131
  format 88
resource retrieval
  file format 88
  resource exit 80
resource user exit 57
resources
  directory 58
  exits 57
  file 57
  fonts 30
  form definitions 21, 31, 33
  group 57
  inline, processing 18, 60, 141
  library 58
resources *(continued)*
  location 58
  overlays 49, 50
  page definitions 51, 54
  page segments 56
  RESTYPE parameter 60
  types of 60
  user-defined 68
restrictions
  PDF indexer 224
RESTYPE
  required by ACIF to process AFP 130
RESTYPE parameter
  flags and values 60

S

Set Coded Font Local structured field 55
skipping header pages 40, 234
SOSI 55
SPCMPRS
  EXTENSIONS parameter 18
specifying application group 173
specifying temporary file for work space 173
STARTINDEXINGONPAGE parameter
  OS/400 indexer 188
STARTTRANSACTIONFIELDSONLINE parameter
  OS/400 indexer 189
STARTTTRIGGERSONLINE parameter
  OS/400 indexer 189
statement date field page
  defining 121
statement number field page
  defining 121
STRMONOND command 247
structured fields
  AFP 87
  Begin Document 94
  Begin Document Index 91
  Begin Named Group 93, 95
  Begin Page 95
  Begin Print File 96
  Begin Resource 89
  Begin Resource Group 88
  Composed Text Control (obsolete) 96
  End Document 95
  End Document Index 93
  End Named Group 93, 95
  End Page 95
  End Print File 96
  End Resource 89
  End Resource Group 89
  group level 90
  Index Element 91, 92, 139
  index object file
    archiving considerations 139
  Invoke Medium Map 43, 139
  Map Coded Font Format 1 96
  Map Coded Font Format 2 45, 96
  page level 90
  Presentation Text Data Descriptor 96
  Set Coded Font Local 55
  Tag Logical Element 87, 92, 93, 95, 139
  Tag Logical Element (TLE) 87

subtotal
  locating 120
subtotal field page
  defining 122
SYSIN JCL statement
  z/OS 131
SYSPRINT JCL statement
  z/OS 131
system date 205
system requirements
  Adobe font requirements 223
  fonts 223

T

Tag Logical Element (TLE)
  structured fields 87
Tag Logical Element structured field
  as part of the indexing process 92, 95
  created in the output document file 93
  defined 92
  examples and rules 87
  in named groups
    out-of-storage problem, possible cause 135
    storage problem, possible cause 135
TEMPDIR parameter
  flags and values 238
TIFF images
  indexing with the 390 indexer 145
total income
  locating 119
total income field page
  defining 121
trace facility 245
TRACE parameter
  flags and values 62
tracedd command 245
TRACEDD parameter
  flags and values
    Multiplatform 63
    z/OS platforms 63
tracelevel command 245
Transaction FIELD syntax
  400 indexer 181
transaction fields 100
TRANSLATEPRINTCONTROL parameter
  400 indexer 189
translation reference characters (TRC) 15, 63
TRC 15, 63
TRC parameter
  flags and values 63
trigger field
  syntax 22
Trigger FIELD syntax
  400 indexer 179
TRIGGER parameter
  390 indexer 172
  options and values 64, 190, 239
TRIGGER1
  defining 109, 118
triggers
  390 indexer 172
  400 indexer parameter 190
  ACIF parameter 64
triggers *(continued)*
  defining 109, 118
  displaying 102, 112
  floating 64, 190
  floating and groupname 34
  PDF indexer parameter 239
  recordrange 64, 190
type of income
  locating 120
type of income field
  defining 122

U

unformatted ASCII data
  ACIF formatting of 6
  description 6
  indexing 72
UNIQUEBNGS parameter
  flags and values 67
user accessibility xi
user exit search order 83
user exits
  index 41, 76
  input 42, 71
  output 48
  output record 78
  print file attributes 84
  provided with ACIF 72
  resource 57
  resource, provided with ACIF 80
user programming exit 72
USERAPPL
  z/OS statement 131
USERLIB
  required by ACIF to process AFP 130
USERLIB parameter
  flags and values 68
USERMASK parameter
  flags and values 69, 70
using ACIF
  in the z/OS environment 131

V

value 205
View Information page
  description 98, 108, 113, 117
  overview 129

X

x,y coordinate system 214
XML indexer
  about 259, 260
  arsload 260
  introduction 259
  invocation 260
  odxmlidx.xsd file 259
  overview 259
  resources 260
  schema 259
  using 259, 260

XML schema 259

Z

z/OS
ACIF parameters 132
ACIF requirements 3, 131
concatenation example 134
DD statement for document file 131
INDEX JCL statement 131
index object file 131
input 131
invoking ACIF 131
JCL example 133, 134
JCL for ACIF job 131
JCL statement 131
JCL to invoke ACIF 131
message file, ACIF 131
OUTPUT JCL statement 131
parameters, ACIF 132
RESOBJ statement 131
SYSIN JCL statement 131
SYSPRINT JCL statement 131
USERAPPL statement 131
using ACIF 131

Product Number: 5724-J33
5697-CM1
5770-RD1

SC19-3354-04