IBM Content Manager OnDemand for iSeries Common Server

Indexing Reference

Version 5 Release 3

SC27-1160-02

IBM Content Manager OnDemand for iSeries Common Server

Indexing Reference

Version 5 Release 3

SC27-1160-02

Note
Before using this information and the product it supports, read the information in "Notices" on page 59.

Third Edition (May 2004)

This edition applies to IBM Content Manager OnDemand for iSeries Version 5 Release 3 and to all subsequent releases and modifications until otherwise indicated in new editions. This edition replaces SC27-1160-01.

Contents

About IBM Content Manager OnDemand for iSeries Common Server Indexing Reference (SC27-1160)

This book contains information about indexing methods, preparing index data, and using tools to index reports that you plan to store in and retrieve from IBM® Content Manager OnDemand for iSeries™ Common Server Version 5 Release 3 (OnDemand).

Who should read this book

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

How this book is organized

This book is organized in the following parts. Each part contains information about one of the indexing tools provided with OnDemand:

- Part 1, "OS/400 indexer reference," on page 1 explains how to use the administrative client graphical tool to define the index criteria that the OS/400® indexer uses to locate and create index data for your spooled files.
- Part 2, "PDF indexer reference," on page 13 describes how to use the OnDemand PDF Indexer to generate index data for Adobe PDF files
- Part 3, "Generic indexer reference," on page 47 describes how to use the OnDemand Generic Indexer to specify index data for other types of input data

Prerequisite and related information

Use the IBM iSeries Information Center as your starting point for looking up iSeries technical information.

You can access the Information Center two ways:

- From the following Web site: `http://www.ibm.com/eserver/iseries/infocenter`
- From CD-ROMs that ship with your Operating System/400® order:

  iSeries Information Center, SK3T-4091-04. This package also includes the PDF versions of iSeries manuals, *iSeries Information Center: Supplemental Manuals*, SK3T-4092-01, which replaces the Softcopy Library CD-ROM.

The Information Center contains advisors and important topics such as Java™, TCP/IP, Web serving, secured networks, logical partitions, clustering, CL commands, and system application programming interfaces (APIs). It also includes links to related IBM Redbooks™ and Internet links to other IBM Web sites such as the IBM home page.

Other information available on the World Wide Web

More iSeries information is available on the World Wide Web. You can access general information from the iSeries home page, which is at the following Web site: `http://www-1.ibm.com/servers/eserver/iseries/`

To access workshops on advanced iSeries functions, use the Technical Studio, located at: `http://www.iseries.ibm.com/tstudio/`

Worldwide, you can read about, select, order and take delivery of iSeries program temporary fixes (PTF) over the Internet. iSeries Internet PTFs (downloads) and Preventive Service Planning (PSP) information are available at the following Internet location: `http://as400service.ibm.com`

iSeries Navigator

IBM iSeries Navigator is a powerful graphical interface for managing your iSeries servers. iSeries Navigator functionality includes system navigation, configuration, planning capabilities, and online help to guide you through your tasks. iSeries Navigator makes operation and administration of the server easier and more productive and is the only user interface to the new, advanced features of the OS/400. It also includes Management Central for managing multiple servers from a central system.

You can find more information on iSeries Navigator in the IBM iSeries Information Center and at the following Web site:
`http://www.ibm.com/eserver/iseries/navigator/`

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. Please send any comments that you have about this publication.

- If you prefer to send comments by FAX, use either of the following numbers:
  - United States, Canada, and Puerto Rico: 1–800–937–3430
  - Other countries: 1–507–253–5192
- If you prefer to send comments electronically, use one of these e-mail addresses:
  - Comments on books: RCHCLERK@us.ibm.com
  - The publication number of a book
  - The page number or topic of a book to which your comment applies

Summary of changes

This edition of *IBM Content Manager OnDemand for iSeries Common Server: Indexing Reference* contains new technical information. There may be some instances where changes were made, but change bars are missing. Significant changes to note are:

- At Version 5 Release 1, Content Manager OnDemand for iSeries (OnDemand) introduced a new server implementation known as the OnDemand Common Server. The Common Server provides enhanced indexing, searching, viewing, security, PDF, and web enablement capabilities for OnDemand users and administrators. Current OnDemand customers who have implemented Spool File Archive (with or without AnyStore or the existing Server feature) can now migrate to the new Common Server using the instructions outlined in Appendix A of the Content Manager OnDemand for iSeries Common Server Planning and Installation Guide. Note that, throughout the documentation, reference to the migration of Spool File Archive data also implies AnyStore data as well, if AnyStore is installed.
- Significant additions have been made to the Content Manager OnDemand for iSeries Common Server Indexing Reference publication regarding functions supported by the OS/400 Indexer. These additions include topics related to defining multi-key indexes, transaction fields, text search fields, SCS spooled files with AFP overlays, and masks for application fields.
- Content Manager OnDemand for iSeries now supports the new iSeries-supported Plasmon optical libraries.
- Two command parameters for the Start Archived Storage Management for OnDemand (STRASMOND) command have been removed to make the use of the command simpler. See Appendix A of the Content Manager OnDemand for iSeries Common Server Administration Guide for details.
- OS/400 has withdrawn the original HTTP server support. In conjunction with this, the Content Manager OnDemand Web Enablement Kit (ODWEK) support for the original HTTP server has also been withdrawn. The HTTP Apache server is now the only supported HTTP server for ODWEK.

Part 1. OS/400 indexer reference

This part provides information about the OS/400 indexer. You can use the OS/400 indexer to specify indexing parameters for SCS, SCS-extended, Advanced Function Presentation™ (AFP™), and Line spooled files that you want to store in the system.

Chapter 1. Using the OS/400 indexer

The OS/400 indexer is the most common OnDemand indexer used for OS/400 spooled files. The OS/400 indexer is called by the ADDRPTOND command for SCS, SCS-extended, Advanced Function Presentation (AFP), and Line spooled files. You use the OnDemand administrative client's graphical indexing tool to define the index criteria that the OS/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 an OnDemand application definition

OnDemand will use the OS/400 indexer by default for SCS, SCS-extended, AFP, and Line spooled files. See the Report Wizard section in the Introduction of the *IBM Content Manager OnDemand for iSeries 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 iSeries Common Server: Administration Guide* for more information on defining an application without using the Report Wizard.

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 new keyword would be added to the OnDemand Application definition. Go to the Indexer Information tab, then click on Keyboard and then Modify to edit the Application's Indexer Parameters. Note that this new keyword ALLOWMULTIPLEVALUES is only valid when BREAK=NO. Also note that unlike the 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

An example

The following example demonstrates how to define a multi-key index 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 top left to bottom right of the report. This has the added benefit of making your indexer parameters easier to understand.

Figure 1 shows a page from the sample report.

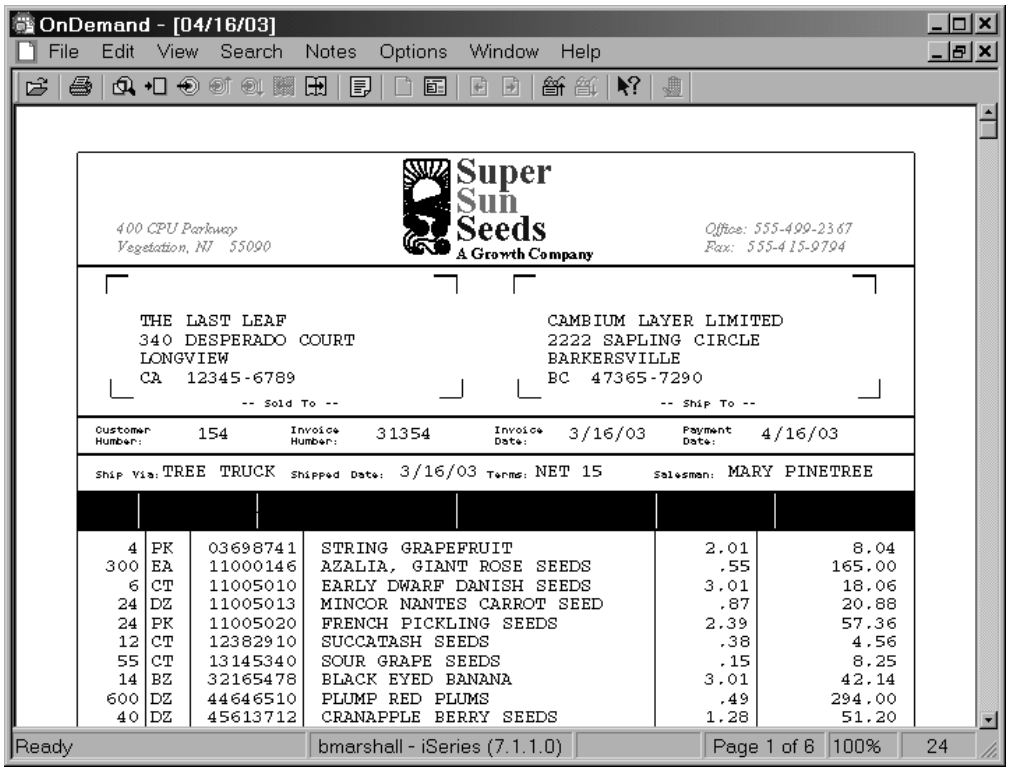


Figure 1. Multi-key index sample report

To begin, first start the 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 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).

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.

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.

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.

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 tab, 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 below).

The fifth field and index are for the total due. The total due is located by using Trigger3. On the Database Field Attributes tab, total due is defined as a string data type.

That completes defining the fields and the indexes.

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.

Click on Index 4. Then click on the Properties button to open the Update an Index dialog box.

Click on the Allow Multiple Values check box. **Note:** This requires Version 7.1.0.8 or later of the OnDemand administrative client.

Click on the OK button to save the item number index as a multi-key index.

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.

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 indexer parameters that were generated for the example report are shown in Figure 2.

```
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*/
```

Figure 2. Multi-key index Indexer Parameters

After loading the example report, you can start the 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. 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 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 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.

An 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 an 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 top left to bottom right of the report. This has the added benefit of making your indexer parameters easier to understand.

A sample page of the report is shown in Figure 3.

```
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
```

Figure 3. Transaction Field Sample Report

To begin, first start the 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 second 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 using internally to 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 indexer parameters that were generated for the example report are shown in Figure 4.

```
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 */
```

Figure 4. Transaction Field Indexer Parameters

After archiving the example report, you can start the OnDemand client, open the new folder, and search for documents.

Defining text search fields

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.

Notes:

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 iSeries 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 in an OnDemand folder definition, follow these steps:

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. Change the name of the new folder.
4. 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.
5. Click OK to save the new folder.

If you prefer, you can delete the folder that was created by the Report Wizard, and always use the new folder that you created to contain the Text Search field. 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 OnDemand 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 OS/400) installed. If PASE is not installed, you will receive message 161 in the 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.

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 OS/400 to put the data into spool as *AFPDS, which is the most efficient way for 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 it is not possible to change the original printer file's DEVTYPE parameter to *AFPDS, 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 OS/400 do it using the DEVTYPE parameter of the printer file as described above.) To enable this conversion, simply specify the data type in the OnDemand Application definition as AFP rather than SCS. When OnDemand encounters an *SCS spooled file that has an overlay, and the Application definition specifies AFP as the data type, OnDemand will convert the *SCS data to *AFPDS and store that newly-created *AFPDS spooled file. Reprints out of OnDemand will require an AFP-capable printer, but that should be expected due to the overlay. Note that if you specify a data type of AFP in your OnDemand Application definitions for any other type of non-AFP spooled file, the loading of the data will fail.

Using a mask when defining applications fields

A mask specifies the pattern of symbols that the indexing program matches with data located for a particular field. With the OS/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='####.##')
```

Note: You may need to manually add the MASK keyword to the correct field definition if you are using a group trigger-based field. Support for group trigger-based field masks may not be available with the graphical indexing tool for the version of the OnDemand administrative client that you are using. Support for float trigger-based field masks was added in Version 7.1.0.6 version of the administrative client.

Part 2. PDF indexer reference

This part provides information about the OnDemand PDF indexer. You can use the PDF indexer to specify indexing parameters for Adobe PDF input files that you want to store in the system.

Chapter 2. Overview

What is the PDF indexer?

The OnDemand PDF indexer is a program that you can use to extract index data from and generate index data about Adobe PDF input files. The index data can enhance your ability to store, retrieve, and view documents with OnDemand. The PDF indexer supports PDF Version 1.3 input and output data streams. For more information about the PDF data stream, see the *Portable Document Format Reference Manual*, published by Adobe Systems Incorporated. Adobe also provides online information with the Acrobat Exchange and Acrobat Distiller products, including online guides for Adobe Capture, PDFWriter, Distiller, and Exchange.

You define and store PDF documents on the server using standard OnDemand functions. You must define an OnDemand application and application group. As part of the application, you must define the indexing parameters used by the PDF indexer to process input files. 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 Command Reference appendix of the *IBM Content Manager OnDemand for iSeries Common Server: Administration Guide* for more information on the ADDRPTOND and STRMONOND commands. See the API Reference appendix of the *IBM Content Manager OnDemand for iSeries Common Server: Administration Guide* for more information on the ARSLOAD API program and its parameters.

After you index and store input files in OnDemand, you use the OnDemand client program to view the PDF document or documents created during the indexing and loading process. You can also print pages of the PDF document you are viewing from the OnDemand client program.

Figure 5 on page 16 illustrates the process of indexing and loading PDF input files.

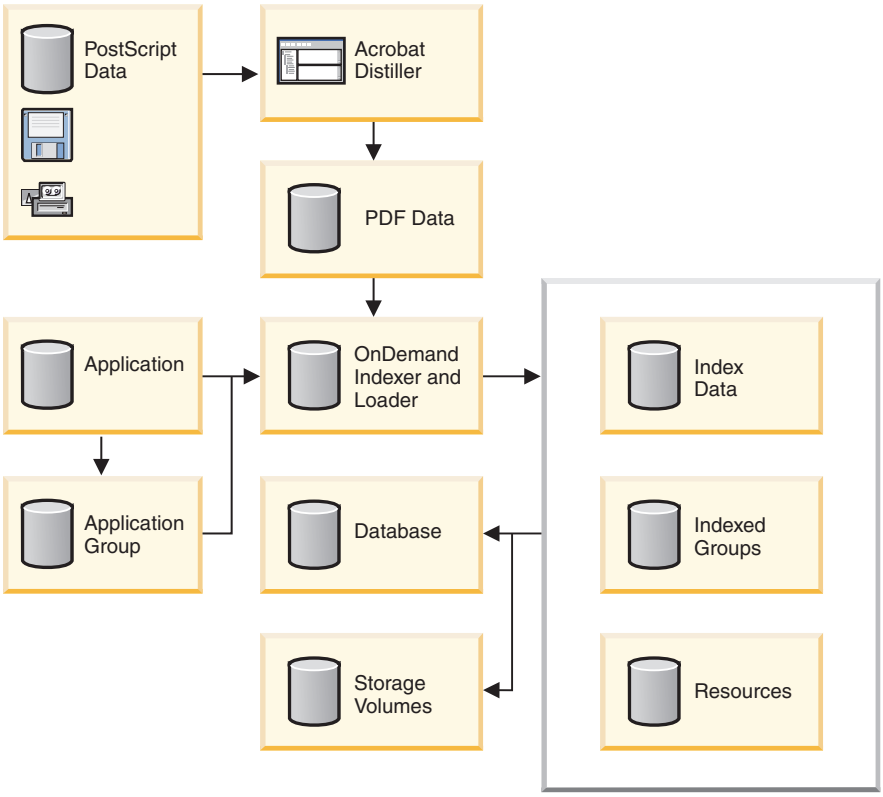


Figure 5. Processing PDF input files in OnDemand

The PDF indexer processes PDF input files. A PDF file is a distilled version of a PostScript file, adding structure and efficiency.

OnDemand 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 OnDemand processes a PDF input file and the application Indexing Information page 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. After indexing the data, OnDemand stores the index data in the database and the indexed groups on disk or archive storage volumes.

How OnDemand uses index information

Every item stored in OnDemand is indexed with one or more *group-level* indexes. Groups are determined when the value of an index changes (for example, account number). When you load a PDF file into the system, OnDemand invokes the PDF indexer to process the indexing parameters and create the index data. OnDemand then loads the index data into the database, storing the group-level attribute values that the PDF indexing program extracted from the data into their corresponding database fields. Figure 6 illustrates the index creation and data loading process.

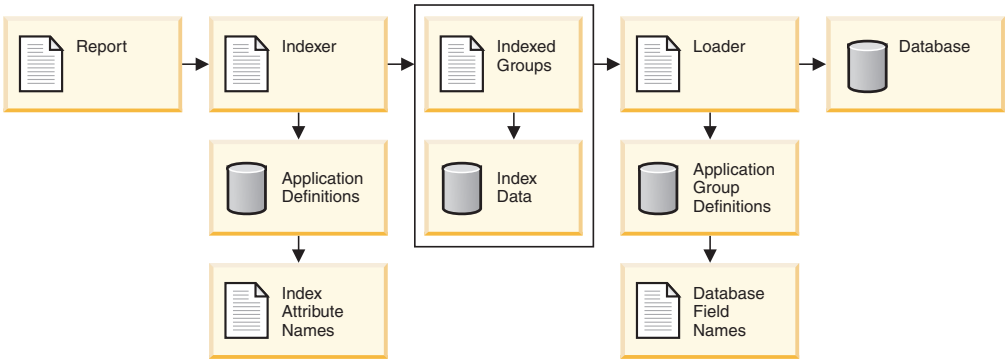


Figure 6. Indexing and loading data

You typically create an application for each report that you plan to store in OnDemand. When you create an application, you define the indexing parameters that the indexing program uses to process the report and create the index data that is loaded into the database. For example, an INDEX parameter includes an attribute name and identifies the FIELD parameter that the indexing program uses to locate the attribute value in the input data. When you create an application, you must assign the application to an application group. The attribute name you specify on an INDEX parameter should be the same as the name of the application group database field into which you want OnDemand to store the index values.

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

To search for reports stored in OnDemand, the user opens a folder. The search fields that appear when the user opens the folder are mapped to database fields in an application group (which, in turn, represent index attribute names). The user constructs a query by entering values in one or more search fields. OnDemand searches the database for items that contain the values (index attribute values) that match the search values entered by the user. Each item contains group-level index information. OnDemand lists the items that match the query. When the user selects an item for viewing, the OnDemand client program retrieves the selected item from disk 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 PDF input files.

Important: If you plan to use the Report Wizard or the graphical indexer to process PDF input files, then you must first install Adobe Acrobat on the PC from which you plan to run the administrative client. You must purchase Adobe Acrobat from Adobe or some other software vendor.

OnDemand provides the `ARSPDF32.API` file to enable PDF viewing from the client. If you install the client after you install Adobe Acrobat, then the installation program will copy the API file to the Acrobat plug-in directory. If you install the client before you install Adobe Acrobat, then you must copy the API file to the Acrobat plug-in directory. Also, if you upgrade to a new version of Acrobat, then you must copy the API file to the new Acrobat plug-in directory. The default location of the API file is `\Program Files\IBM\OnDemand32\PDF`. The default Acrobat plug-in directory is `\Program Files\Adobe\Acrobat x.y\Acrobat\Plug_ins`, where `x.y` is the version of Acrobat, for example, `4.0`, `5.0`, and so forth.

Beginning with Version 5.2, you can define indexing information in a visual environment. You begin by opening a sample input file with the graphical indexer. (**Note:** The input file is limited to a PC file when using the graphical PDF indexer. The graphical PDF indexer is designed to work with workstation PDF files, not PDF spooled files in an output queue on the iSeries server.) You can run the graphical indexer from the report wizard or by choosing the sample data option from the Indexing Information page of the application. After you open an input file in the graphical indexer, you define triggers, fields, and indexes. The PDF indexer uses the triggers, fields, and indexes to locate the beginning of a document in the input data and extract index values from the input data. Once you have defined the triggers, fields, and indexes, you can save them in the application so that OnDemand can use them later on to process the input files that you load into the system.

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

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

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

1. To begin, start the administrative client.
2. Log on to a server.
3. Start the report wizard by clicking the Report Wizard icon on the toolbar. The report wizard opens the Sample Data dialog box.
4. Click Select Sample Data to open the Open dialog box. **Note:** The Sample Data is limited to a PC file when using the graphical PDF indexer. The graphical PDF indexer is designed to work with workstation PDF files, not PDF spooled files in an output queue on the iSeries server.
5. Type the name or full path name of a file in the space provided or use the Look in or Browse commands to locate a file.
6. Click Open. The graphical indexer opens the input file in the report window.
7. Press F1 to open the main help topic for the report window. The main help topic contains general information about the report window and contains links to other topics that describe how to add triggers, fields, and indexes. Under Options and Commands, click Indexer Information page to open the Indexing Commands topic. (You can also use the content help tool to display information about the icons on the toolbar.) Under Tasks, Indexer Information page, click Adding a trigger (PDF).
8. Close any open help topics and return to the report window.
9. Define a trigger.
   - Find a text string that uniquely identifies the beginning of a document. For example, `Account Number`, `Invoice Number`, `Customer Name`, and so forth.
   - Using the mouse, draw a box around the text string. Start just outside of the upper left corner of the string. Click and hold 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 icon 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, double click the trigger. The graphical indexer highlights the text string in the current document. Double click the trigger again. The graphical indexer should highlight the text string on the first page of the next document. Use the Select dialog box to move forward to the first page of each document and return to the first document in the input file.
   - Put the report window in add mode.
10. Define a field and an index.
    - Find a text string that can be used to identify the location of the field. The text string should contain a sample index value. For example, if you want to extract account number values from the input file, then find where the account number is printed on the page.
    - Using the mouse, draw a box around the text string. Start just outside of the upper left corner of the string. Click and hold 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 icon on the toolbar to open the Add a Field dialog box.
- On the Field Information page, verify the attributes of the index field. For example, the text string that you selected in the report window should be displayed under Reference String; the Trigger should identify the trigger on which the field is based. Click Help for assistance with the options and values that you can specify.
- On the Database Field Attributes page, verify the attributes of the database field. In the Database Field Name space, enter the name of the application group field into which you want OnDemand to store the index value. In the Folder Field Name space, enter the name of the folder field that will appear on the client search screen. Click Help for assistance with the other options and values that you can specify.
- Click OK to define the field and index.
- To verify the locations of the fields, first put the report window in display mode. The fields should have a blue box drawn around them. Next, click the Select tool to open the Select dialog box. Under Fields, double-click Field 1. The graphical indexer highlights the text string in the current document. Double click Field 1 again. The graphical indexer should move to the next document and highlight the text string. Use the Select dialog box to move forward to each document and display the field. Then return to the first document in the input file.
- Put the report window in add mode.

11. Click the Display Indexer Parameters tool to open the Display Indexer Parameters dialog box. The Display Indexer Parameters dialog box lists the indexing parameters that the PDF indexer will use to process the input files that you load into the application. At a minimum, you need one trigger, one field, and one index. See Chapter 4, “Parameter reference,” on page 27 for details about the indexing parameters.
12. When you have finished defining all of the triggers, fields, and indexes, close the report window.
13. Click Yes to save the changes to the indexer parameters.
14. On the Sample Data window, click Next to continue with the report wizard.

Manually indexing input data

Note: If you prefer creating your own PDF indexing parameters manually rather than using the graphical PDF indexer, you can use the instructions in the remainder of this chapter to do so.

Indexing concepts

Indexing parameters include information that allow the PDF indexer to identify key items in the print data stream, *tag* these items, and create *index elements* pointing to the tagged items. OnDemand uses the tag and index data for efficient, structured search and retrieval. You specify the index information that allows the PDF indexer to segment the data stream into individual items, called *groups*. A group is a collection of one or more pages, such as a bank statement, insurance policy, phone bill, or other logical segment of a report. The PDF indexer creates indexes for each group when the value of an index changes (for example, account number).

A tag is made up of an *attribute name*, for example, Customer Name, and an *attribute value*, for example, Earl Hawkins. Tags also include information that tell the PDF indexer where to locate the attribute value on a page. For example, a tag used to collect customer name index values provides the PDF indexer with the starting and ending position on the page where the customer name index values appear. The PDF indexer generates index data and stores it in a generic index file.

Coordinate system

The location of the text strings the PDF indexer uses to determine the beginning of a group and index values are described as *x* and *y* pairs in a coordinate system imposed on the page. For each text string, you identify its upper left and lower right position on the page. The upper left corner and lower right corner form a string box. The string box is the smallest rectangle that completely encloses the text string. The origin is in the upper left hand corner of the page. The *x* coordinate increases to the right and *y* increases down the page. You also identify the page on which the text string appears. For example, the text string Customer Name, that starts 4 inches to the right and 1 inch down and ends 5.5 inches to the right and 1.5 inches down on the first page in the input file can be located as follows:

```
ul(4,1),lr(5.5,1.5),1,'Customer Name'
```

OnDemand provides the ARSPDUMP command to help you identify the locations of text strings on the page. See Chapter 7, "ARSPDUMP reference," on page 43 for more information about ARSPDUMP.

Indexing parameters

Processing parameters can contain index and conversion parameters, options, and values. For most reports, the PDF indexer requires at least three indexing parameters to generate index data:

- TRIGGER

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

  – The PDF indexer compares words in the input file with the text string specified in a trigger.
  – The location of the trigger string value must be identified using the *x,y* coordinate system and page offsets.
  – A maximum of 16 triggers can be specified.
  – All triggers must match before the PDF indexer can begin to locate index information.

- FIELD

  The field parameter specifies the location of the data that the PDF indexer uses to create index values.

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

- INDEX

The index parameter is where you specify the attribute name and identify the field or fields on which the index is based. We strongly encourage you to name the attribute the same as the application group database field name.

- The PDF indexer creates indexes for a group of one or more pages.
- You can concatenate field parameters to form an index.
- A maximum of 32 index parameters can be specified.

The PDF indexer creates a new group and extracts new index values when one or more of the index values change.

Figure 7 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.

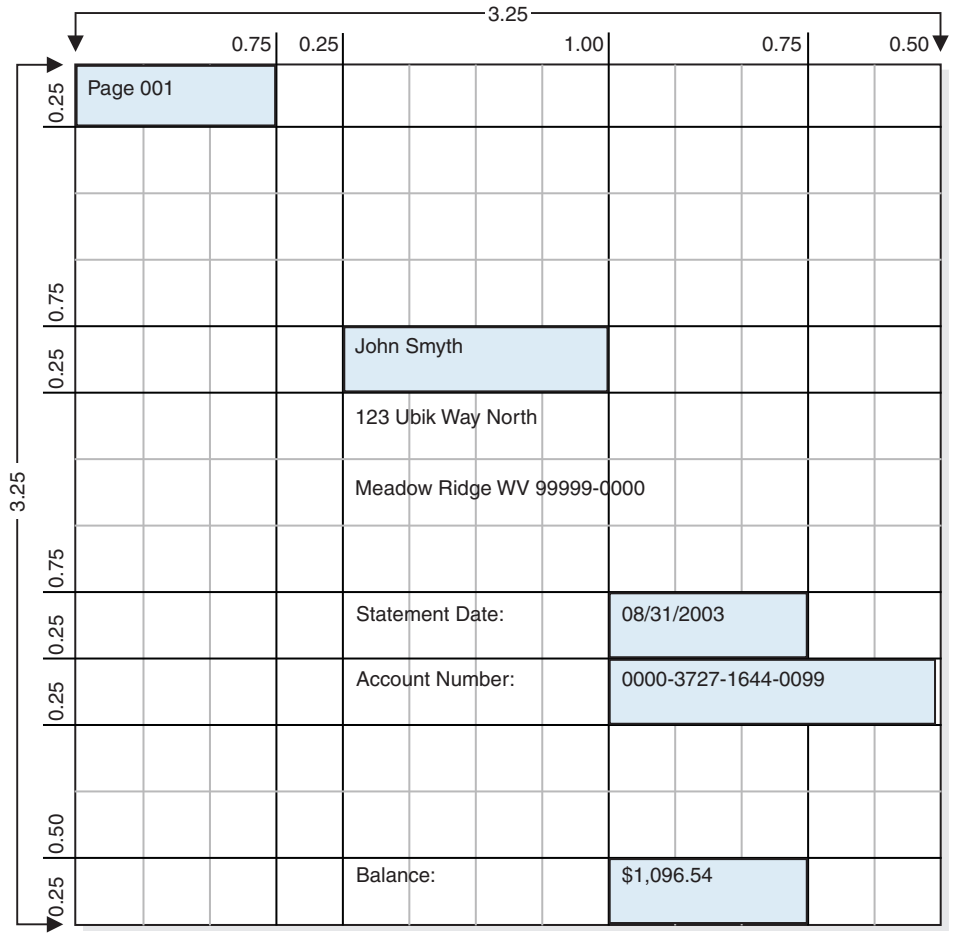


Figure 7. Indexing data with the PDF indexer

TRIGGER parameters tell the PDF indexer how to identify the beginning of a group in the input. The PDF indexer requires one TRIGGER parameter to identify the beginning of a group (statement) in the sample file. FIELD parameters determine the location of index values in a statement. Fields are based on the location of trigger records. INDEX parameters identify the attribute names of the index fields. Indexes are based on one or more field parameters. The following parameters could be used to index the report depicted in Figure 7. See Chapter 4, "Parameter reference," on page 27 for details about the parameter syntax.

- Define a trigger to search each page in the input data for the text string that identifies the start of a group (statement):

  ```
  TRIGGER1=ul(0,0),lr(.75,.25),*,'Page 001'
  ```

- Define fields to identify the location of index data. For the sample report, we might define four fields:
  - FIELD1 identifies the location of customer name index values.

    ```
    FIELD1=ul(1,1),lr(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
    ```

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.

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). See Chapter 7, "ARSPDUMP reference," on page 43 for more information about ARSPDUMP.

The process works as follows:

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

- Create TRIGGER and FIELD parameters using the *x,y* coordinates, page number, and string value.

Indexing parameters are part of the OnDemand application. The administrative client provides an edit window you can use to maintain indexing parameters for the application.

Chapter 3. System considerations

System limitations

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

- The PDF indexer can process PDF input files that are up to 4 GB in size
- IBM recommends that the CCSID of the PDF input file be 1252 (WinAnsiEncoding). Using another CCSID may cause unexpected results.
- The PDF indexer supports DBCS languages. However, IBM does not provide any DBCS fonts. You can purchase DBCS fonts from Adobe. The PDF indexer supports all DBCS fonts, except encrypted Japanese fonts.
- Input data delimited with PostScript Passthrough markers cannot be indexed
- The Adobe Toolkit does not validate link destinations or bookmarks to other pages in a document or to other documents. Links or bookmarks may or may not resolve correctly, depending on how you segment your documents.
- If a font is referenced in an input file but not embedded in the file and the PDF indexer cannot locate the font, the referenced font is substituted by using one of the base Adobe Type 1 fonts that are provided by IBM. If the customer purchases additional fonts and installs them on the system, the additional fonts can be embedded at indexing time if they are referenced in an input file and the location is specified on the FONTLIB parameter. See “FONTLIB” on page 30 for more information.

Input data requirements

The PDF indexer processes PDF input data. PostScript data generated by applications must be processed by Acrobat Distiller before you run the PDF indexer. The online documentation provided with Acrobat Distiller describes methods you can use to generate PDF data.

If you plan to automate the data indexing and loading process on the OnDemand server, the input file name must identify the application group and application to load. Use the following convention to name 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 filename to identify the application group to load. However, you can use the **-G** parameter to specify a different part of the filename (MVS™, JOBNAME, or DATASET) that identifies the application group to load. For example, `arsload -G JOBNAME`.

If the application group contains more than one application, you must identify the application to load; otherwise the load will fail. 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. For example, `arsload -A DATASET`.

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

NLS considerations

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

Data values that you specify on TRIGGER and FIELD parameters must be encoded in the same code page as the document. For example, if the characters in the document are encoded in code page 1252, any data values that you specify on TRIGGER and FIELD parameters must be encoded in code page 1252. Examples of data values that you might specify include TRIGGER string values and FIELD default and constant values.

When loading data using the PDF indexer, the locale must be set appropriately for the code page of the documents. For example, if the code page of the documents is 954, set the locale environment variable to ja_JP or some other locale that correctly identifies upper and lower case characters in code page 954.

For more information about NLS in OnDemand, see the *IBM Content Manager OnDemand for iSeries Common Server Planning and Installation Guide*.

Chapter 4. Parameter reference

This parameter reference assumes that you will use the ARSLOAD program to process your input files. When you use the ARSLOAD program to process input files, the PDF indexer ignores any values that you may provide for the INDEXDD, INPUTDD, MSGDD, OUTPUTDD, and PARMDD parameters. If you run the ARSPDOCI program from the command prompt or call it from a user-defined program, then you must provide values for the INPUTDD, OUTPUTDD, and PARMDD parameters and verify that the default values for the INDEXDD and MSGDD parameters are correct.

COORDINATES

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

Required?
No

Default Value
IN

Syntax

COORDINATES=*metric*

Options and values

The *metric* can be:

IN

The coordinate metrics are specified in inches (the default).

CM

The coordinate metrics are specified in centimeters.

MM

The coordinate metrics are specified in millimeters.

FIELD

Identifies the location of index data and can provide default and constant index values. You must define at least one field. You can define up to 32 fields. You can define two types of fields: a *trigger field*, which is based on the location of a trigger string value and a *constant field*, which provides the actual index value that is stored in the database.

Required?
Yes

Default Value
<none>

Trigger field syntax

FIELD*n***=ul(***x,y***),lr(***x,y***),***page***[,(TRIGGER=***n***,BASE={0 | TRIGGER},**
MASK='*field_mask***',DEFAULT='***value***')]**

Options and values

n

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

ul(*x,y***)**

The coordinates for the upper left corner of the field string box. The field string box is the smallest rectangle that completely encloses the field string value (one or more words on the page). The PDF indexer must find the field string value inside the field string box. The supported range of values is 0 (zero) to 45, page width and length, in inches.

lr(*x,y***)**

The coordinates for the lower right corner of the field string box. The field string box is the smallest rectangle that completely encloses the field string value (one or more words on the page). The PDF indexer must find the field string value inside the field string box. The supported range of values is 0 (zero) to 45, page width and length, in inches.

page

The sheet number where the PDF indexer begins searching for the field, relative to a trigger or 0 (zero) for the same page as the trigger. If you specify BASE=0, 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)`.

Note: Beginning with Version 5.2, 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, so long as the resulting absolute field coordinates of the field string rectangle are still in the range of `0 <= x <= 45` and `0 <= y <= 45`. The `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. 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.

Note: The string that you specify for the mask can contain any character. For example, given the following definitions:

```
TRIGGER2=*,25,'ACCOUNT'
FIELD2=0,38,11,(TRIGGER=2,BASE=0,MASK='@000-####-#')
```

The PDF indexer selects the field only if the data in the field columns contains an eleven-character string comprised of any letter, three zeros, a dash character, any four numbers, a dash character, and any number.

DEFAULT='*value***'**

Defines the default index value, when there are no words within the coordinates provided for the field string box.

For example, assume that an application program generates statements that contain an audit field. The contents of the field can be PASSED or FAILED. However, if a statement has not been audited, the application program does not generate a value. In that case, there are no words within the field string box. To store a default value in the database for unaudited records, define the field as follows:

```
FIELD3=ul(8,1),lr(8.5,1.25),1,(DEFAULT='NOT AUDITED')
```

The PDF indexer assigns the index associated with FIELD3 the value `NOT AUDITED`, if the field string box is blank.

Examples

The following field parameter causes the PDF indexer to locate the field at the coordinates provided for the field string box. The field is based on TRIGGER1 and located on the same page as TRIGGER1. Specify BASE=0 because the field string box always appears in a specific location on the page.

```
TRIGGER1=ul(0,0),lr(.75,.25),*,'Page 0001'
FIELD1=ul(1,1),lr(3.25,1.25),0,(TRIGGER=1,BASE=0)
```

Constant field syntax

FIELD*n*=*'constant'*

Options and values

n

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

'constant'

The literal (constant) string value of the field. This is the index value stored in the database. The constant value can be 1 to 250 bytes in length. The PDF indexer does not validate the type or content of the constant.

Examples

The following field parameter causes the PDF indexer to store the same text string in each INDEX1 value it creates.

```
FIELD1='000000000'
INDEX1='acct',FIELD1
```

The following field parameters cause the PDF indexer to concatenate a constant value with the index value extracted from the data. The PDF indexer concatenates the constant value specified in the FIELD1 parameter to each index value located using the FIELD2 parameter. The concatenated string value is stored in the database. In this example, the account number field in the data is 14 bytes in length. However, the account number in the database is 19 bytes in length. Use a constant field to concatenate a constant five byte prefix (0000–) to all account numbers extracted from the data.

```
FIELD1='0000-'
FIELD2=ul(2,2),lr(2.5,2.25),0,(TRIGGER=1,BASE=0)
INDEX1='acct_num',FIELD1,FIELD2
```

Related parameters

INDEX parameter on page 31.
TRIGGER parameter on page 36.

FONTLIB

Identifies the directory or directories in which fonts are stored. Specify any valid path. The PDF indexer searches for fonts in the order that the paths are listed. If a font is referenced in an input file but not embedded in the file, the PDF indexer attempts to locate the font in the directory or directories listed on the FONTLIB parameter. If the font is located, the PDF indexer adds it to the output file. If a font is referenced in an input file and the PDF indexer cannot locate the font, the referenced font is substituted by using one of the base Adobe Type 1 fonts that are provided by IBM. If the customer purchases additional fonts and installs them on the system, the additional fonts can be embedded at indexing time if they are referenced in an input file and are present in one of the directories specified on the FONTLIB parameter.

Required?
No

Default Value
/QIBM/ProdData/OnDemand/Adobe/fonts

Syntax

FONTLIB=*pathlist*

Options and values

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

```
/QIBM/ProdData/OnDemand/Adobe/fonts:/mycustom/fonts
```

The PDF indexer searches the paths in the order in which they are specified. Delimit path names with the colon (:) character.

INDEX

Identifies the index name and the field or fields on which the index is based. You must specify at least one index parameter. You can specify up to 32 index parameters. When you create index parameters, IBM recommends that you name the index the same as the application group database field name.

Required?
Yes

Default Value
<none>

Syntax

INDEXn=*'name'*,FIELDnn[,...FIELDnn]

Options and values

n

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

'name'

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

The index name is a string from 1 to 250 bytes in length. We strongly encourage you to name the index the same as the application group database field name.

FIELD*nn*

The name of the field parameter or parameters that the PDF indexer uses to locate the index. You can specify a maximum of 32 field parameters. Separate the field parameter names with a comma. The total length of all the specified field parameters cannot exceed 250 bytes.

Examples

The following index parameter causes the PDF indexer to create group-level indexes for date index values (the PDF indexer supports group-level indexes only). When the index value changes, the PDF indexer closes the current group and begins a new group.

```
INDEX1='report_date',FIELD1
```

The following index parameters cause the PDF indexer to create group-level indexes for customer name and account number index values. The PDF indexer closes the current group and begins a new group when either the customer name or the account number index value changes.

```
INDEX1='name',FIELD1
INDEX2='acct_num',FIELD2
```

Related parameters

FIELD parameter on page 27.

INDEXDD

Determines 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 PDF indexer ignores any value that you may supply for the INDEXDD parameter. If you process input files with the ARSPDOCI program, then verify the value of the INDEXDD parameter.

Default Value
INDEX

Syntax

INDEXDD=*filename*

Options and values

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

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 triggers and fields are found. 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.

Note: When you use INDEXSTARTBY to skip header pages, the PDF indexer does not copy non-indexed pages to the output file or store them in OnDemand. For example, if you specify INDEXSTARTBY=3 and the first group is found on page three, then pages one and two are not copied to the output file or stored in OnDemand. If you specify INDEXSTARTBY=3 and the first group is found on page two, then page one is not copied to the output file or stored in OnDemand.

Required?
No

Default Value
1

Syntax

INDEXSTARTBY=*value*

Options and values

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

INPUTDD

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

Required?
No

Note: When you process input files with the ARSLOAD program, 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.

Default Value
<none>

Syntax

INPUTDD=*name*

Options and values

The *name* is the file name or full path name of the input file. If you specify the file name without a path, the PDF indexer searches the current directory for the specified file.

MSGDD

Determines the name or the full path name of the file where the PDF indexer writes error messages. If you do not specify the MSGDD parameter, the PDF indexer writes messages to the display (interactive) or the joblog (batch).

Required?
No

Note: When you process input files with the ARSLOAD program, the PDF indexer ignores any value that you may supply for the MSGDD parameter. If you process input files with the ARSPDOCI program, then verify the value of the MSGDD parameter.

Default Value
the display (interactive) or the joblog (batch), which are sometimes referred to as stderr (standard error)

Syntax

MSGDD=*name*

Options and values

The *name* is the file name or full path name where the PDF indexer writes error messages. If you specify the file name without a path, the PDF indexer places the error file in the current directory.

OUTPUTDD

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

Required?
No

Note: When you process input files with the ARSLOAD program, the PDF indexer ignores any value that you may supply for the OUTPUTDD parameter. If you process input files with the ARSPDOCI program, then you must specify a value for the OUTPUTDD parameter.

Default Value
<none>

Syntax

OUTPUTDD=*name*

Options and values

The *name* is the file name or full path name of the output file. If you specify the file name without a path, the PDF indexer puts the output file in the current directory.

PARMDD

Identifies the name or the full path name of the file that contains the indexing parameters used to process the input data.

Required?
No

Note: When you process input files with the ARSLOAD program, the PDF indexer ignores any value that you may supply for the PARMDD parameter. If you process input files with the ARSPDOCI program, then you must specify a value for the PARMDD parameter.

Default Value
<none>

Syntax

PARMDD=*name*

Options and values

The *name* is the file name or full path name of the file that contains the indexing parameters. If you specify the file name without a path, the PDF indexer searches for the file in the current directory.

TEMPDIR

Determines the name of the directory that the PDF indexer uses for temporary work space.

Required?
No

Default Value
/arstmp

Syntax

TEMPDIR=*directory*

Options and values

The *directory* is a valid directory name.

TRACEDD parameter

The TRACEDD parameter is new in Version 5.3. For more information, see Chapter 8, “Trace facility,” on page 45.

TRIGGER

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

Required?
Yes

Default Value
<none>

Syntax

TRIGGERn**=ul(***x,y***),lr(***x,y***),***page***,'***value***'**

Options and values

n

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

ul(*x,y***)**

The coordinates for the upper left corner of the trigger string box. The trigger string box is the smallest rectangle that completely encloses the trigger string value (one or more words on the page). The PDF indexer must find the trigger string value inside the trigger string box. The supported range of values is 0 (zero) to 45, page width and length, in inches.

lr(*x,y***)**

The coordinates for the lower right corner of the trigger string box. The trigger string box is the smallest rectangle that completely encloses the trigger string value (one or more words on the page). The PDF indexer must find the trigger string value inside the trigger string box. The supported range of values are 0 (zero) to 45, page width and length, in inches.

page

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

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

'*value*'

The actual string value the PDF indexer uses to match the input data. The string value is case sensitive. The value is one or more words that can be found on a page.

Examples

TRIGGER1

The following TRIGGER1 parameter causes the PDF indexer to search the specified location on every page of the input data for the specified string. You must define TRIGGER1 and the page value for TRIGGER1 must be an asterisk.

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

Group triggers

The following trigger parameter causes the PDF indexer to attempt to match the string value `Account Number` within the coordinates provided for the trigger string box. The trigger can be found on the same page as TRIGGER1.

```
TRIGGER2=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)
```

Related parameters

The FIELD parameter on page 27.

Chapter 5. Message reference

The PDF indexer creates a message list at the end of each indexing run. A return code of `0` (zero) means that processing completed without any errors.

The PDF indexer detects a number of error conditions that can be logically grouped into several categories:

- **Informational**

  When the PDF indexer processes a 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 8 (eight) or 16 (sixteen) and terminates processing the current input file. Most error conditions detected by the PDF indexer fall into this category. The exact method of termination may vary. For certain severe errors, the PDF indexer may fail with a segment fault. This is generally the case when some system service fails. In other cases, the PDF indexer terminates with the appropriate error messages written either to standard error or to a file. When the PDF indexer is invoked by the ARSLOAD program, error messages are automatically written to the system log. If you run the ARSPDOCI command, you can specify the name or the full path name of the file to hold the processing messages by using the **MSGDD** parameter.
- **Adobe Toolkit**
- **Internal Error**

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

See *IBM DB2® Content Manager OnDemand: Messages and Codes*, SC27-1379 for a list of the messages that may be generated by the PDF indexer, along with explanations of the messages and actions that you can take to respond to the messages. The messages that are generated by the PDF indexer are listed in the Common Server section of the messages publication.

Chapter 6. ARSPDOCI reference

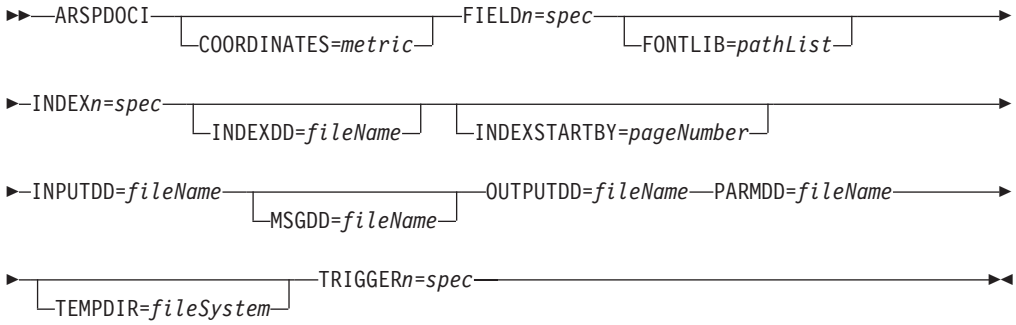
Purpose

Generate index data for a PDF file.

The ARSPDOCI program uses the identified locations of text strings on a page of a PDF document to produce a text index file as well as a byte offset indexed PDF document. You can use the ARSPDUMP program to list the locations of text strings in a document. See Chapter 7, "ARSPDUMP reference," on page 43 for more information.

Syntax

Note: The following syntax should be used only when you run the ARSPDOCI program from the command line or call it from a user-defined program.



Description

The ARSPDOCI program can be used to index a PDF file. The ARSLOAD program automatically calls the ARSPDOCI program if the input data type is PDF and the indexer is PDF. If you need to index a PDF file and you do not want to use the ARSLOAD program to process the file, then you can run the ARSPDOCI program from the command line or call it from a program.

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

If a font is referenced in an input file but not embedded in the file and the PDF indexer cannot locate the font, the referenced font is substituted by using one of the base Adobe Type 1 fonts that are provided by IBM. If the customer purchases additional fonts and installs them on the system, the additional fonts can be embedded at indexing time if they are referenced in an input file and the location is specified on the FONTLIB parameter. See "FONTLIB" on page 30 for more information.

Parameters

Refer to Chapter 4, "Parameter reference," on page 27 for details about the parameters that you can specify when you run the ARSPDOCI program from the command line or a user-defined program.

IFS location

/usr/bin/arspdoci
The executable program.

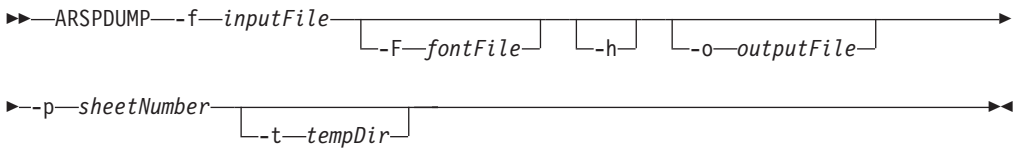
Chapter 7. ARSPDUMP reference

Purpose

Print the locations of text strings on a page.

The ARSPDUMP program lists the locations of text strings on a page in a PDF file. The output of the ARSPDUMP program contains a list of the text strings on the page and the coordinates for each string. You can use the information that is generated by the ARSPDUMP program to create the parameter file that is used by the ARSPDOCI program to index PDF files. See Chapter 6, "ARSPDOCI reference," on page 41 for more information.

Syntax



Description

The ARSPDUMP program can be used to identify the locations of text strings on a page in a PDF file.

The output of the ARSPDUMP program contains a list of the text strings on the page and the coordinates for each string.

If a font is referenced in a PDF file, but not embedded, then the ARSPDUMP program attempts to find the font using information provided with the **-F** parameter. If the ARSPDUMP program does not find the font, then it uses a substitute Adobe Type 1 font.

Parameters

-f inputFile
The file name or full path name of the PDF file to process.

-F fontDir
Identifies directories in which fonts are stored. Specify any valid path. Use the colon (:) character to separate path names. The ARSPDUMP program searches the paths in the order in which they are specified. If you do not specify this flag and name a font directory, then the ARSPDUMP program attempts to locate fonts in the /QIBM/ProdData/OnDemand/Adobe/fonts directory.

-h Lists the parameters and their descriptions for the ARSPDUMP program.

-o outputFile
The file name or full path name of the file into which the ARSPDUMP

program writes output messages. If you do not specify this flag and name a file, then the ARSPDUMP program writes output to the display (interactive) or the joblog (batch).

-p sheetNumber
The number of the page in the PDF file that you want the ARSPDUMP program to process. This is the page that contains the text strings that you want to use to define triggers and fields. The sheet number is the order of the page as it appears in the file, beginning with the number 1 (one), for the first page in the file. Contrast with page identifier, which is user-defined information that identifies each page (for example, iv, 5, and 17-3).

-t tempDir
Identifies the directory that the ARSPDUMP program uses for temporary work space. Specify any valid directory name. If you do not specify this flag and name a directory, then the ARSPDUMP program uses the /arstmp directory for temporary work space.

Examples

The following example shows how to invoke the ARSPDUMP program 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
```

See the *IBM Content Manager OnDemand for iSeries Common Server Administration Guide* for more information about running ARSPDUMP using QSHELL.

IFS location

/usr/bin/arspdump
The executable program.

Chapter 8. Trace facility

Beginning with Version 5.3, an enhanced tracing capability for the PDF indexer is now available. The tracing capability provides assistance to users attempting to debug problems, such as when the system fails during the indexing and loading of PDF documents.

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

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

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

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

Where:

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

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

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

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

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

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

Note: See Chapter 6, "ARSPDOCI reference," on page 41 for more information about the parameters that may be specified when running the ARSPDOCI program.

After running the PDF indexer with the trace, the output file specified by the `tracedd=` parameter will contain detailed information about the processing that took place and where the PDF indexer is failing during the process. The trace information will identify whether a trigger was not found, a field was not found, the PDF data is corrupted, there was a problem extracting a PDF page from the document, or even if there is not enough memory or disk space to complete the required operations. Figure 8 on page 46 shows an example of the trace information that may be generated by the PDF indexer.

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

 .
 .
 .

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

Figure 8. Trace Information for the PDF indexer

Part 3. Generic indexer reference

This part provides information about the OnDemand generic indexer. You can use the generic indexer to specify index data for other types of input files that you want to store in the system. (Input files that do not contain PDF, SCS, SCS-extended, Advanced Function Presentation (AFP), or Line spooled data.)

Chapter 9. Overview

OnDemand provides the generic indexer to allow you to specify indexing information for input data that you cannot or do not want to index with the OS/400 Indexer or the PDF Indexer. For example, suppose that you want to load files into the system that were created by using a word processor. The files can be stored in the system in the same format in which they were created. The files can be retrieved from the system and viewed by using the word processor. However, because the documents do not contain PDF, SCS, SCS-extended, AFP, or LINE spooled data, you cannot index them with the other indexers that are provided with the OnDemand product. You can specify index information about the files in the 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 the OnDemand client program.

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

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

Loading data

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

There are two ways to run the ARSLOAD program:

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

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

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

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

```
MVS.JOBNAME.DATASET.FORM.YYYYDDD.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.

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.

The Generic indexer determines where the AFP resources end in the file and process the documents using the offsets and lengths that you provide, relative to where the resources end.

Chapter 10. Specifying the parameter file

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

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

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

Important:

1. The parameter names in the parameter file are case sensitive and must appear in upper case. For example, `GROUP_FIELD_NAME:account` is valid, while `group_field_name:account` is not.
2. When loading data using the Generic indexer, the locale must be set appropriately for the CODEPAGE: parameter. For example, if `CODEPAGE:954` is specified, set the locale environment variable to `ja_JP` or some other locale that correctly identifies upper and lower case characters in code page 954.

CODEPAGE:

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

Important: When loading data using the Generic indexer, the locale must be set appropriately for the CODEPAGE: parameter. For example, if `CODEPAGE:954` is specified, set the locale environment variable to `ja_JP` or some other locale that correctly identifies upper and lower case characters in code page 954.

Syntax

CODEPAGE:`cpgid`

Options and values

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

The **CODEPAGE:** parameter is required.

Example

The following illustrates how to specify a code page of 37 for the input data:

```
CODEPAGE:37
```

COMMENT:

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

Syntax

COMMENT: `text on a single line`

Options and values

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

Example

The following are examples of comment lines:

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

GROUP_FIELD_NAME:

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

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

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

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

Syntax

GROUP_FIELD_NAME:`applgrpFieldName`

Options and values

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

Example

The following shows examples of application group field names:

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

GROUP_FIELD_VALUE:

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

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

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

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

Syntax

GROUP_FIELD_VALUE:`value`

Options and values

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

Example

The following shows examples of index values:

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

GROUP_FILENAME:

The file name or full path name of the input file. If you do not specify a path, then the generic indexer searches the current directory for the specified file; however, you should always specify the full path name of the input file.

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 program.

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

After successfully loading the data, the system deletes the input file that is specified on the GROUP_FILENAME: parameter if the file name extension is .OUT, and for daemon mode processing, the rest of the input file name is the same as the .ARD file name. The system also deletes the .IND file (the Generic indexer parameter file) and the .ARD file (the dummy file that is used to initiate a load process when the ARSLOAD program is running in daemon mode). See "Loading data" on page 49 for more information.

Syntax

GROUP_FILENAME:`fileName`

Options and values

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

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

Example

The following are valid file name lines:

```
GROUP_FILENAME:/tmp/statements
GROUP_FILENAME:D:\ARSTMP\statements
GROUP_FILENAME:/tmp/ondemand/inputfiles/f1b0a1600.out
GROUP_FILENAME:
```

GROUP_LENGTH:

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

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

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

Syntax

GROUP_LENGTH:`value`

Options and values

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

Example

The following illustrates how to specify length values:

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

GROUP_OFFSET:

Specifies the starting location (byte offset) into the input file of the data to be indexed. Specify `0` (zero) for the first byte (the beginning) of the file. (If processing AFP documents and resources with the Generic indexer, see "Processing AFP data" on page 50.) Each group that you specify in the parameter file must contain one **GROUP_OFFSET:** line. The **GROUP_OFFSET:** line must follow the **GROUP_FIELD_NAME:** and **GROUP_FIELD_VALUE:** lines that comprise a group. For example:

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

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

Syntax

GROUP_OFFSET:`value`

Options and values

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

Example

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

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

Chapter 11. 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:37
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:37
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
```

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only the IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe on any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator
3605 Highway 52 N
Rochester, MN 55901–7829
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

Advanced Function Presentation, AFP, IBM, iSeries, Operating System/400, OS/400, and Redbooks are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Adobe, the Adobe logo, Acrobat, and the Acrobat logo are trademarks of Adobe Systems Incorporated, which may be registered in certain jurisdictions.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Index

H

I

L

M

N

O

P

R

S

T

X

Program Number: 5722-RD1

SC27-1160-02