Fast Path Indexer/EP
User Guide

Supporting
Version 3.11 of Fast Path Indexer/EP

January 2014
Contacting BMC Software

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

Address: BMC SOFTWARE INC
2101 CITYWEST BLVD
HOUSTON TX 77042-2827
USA

Telephone: 1 713 918 8800 or 1 800 841 2031
Fax: 1 713 918 8000

Outside United States and Canada

Telephone: +01 713 918 8800
Fax: +01 713 918 8000


BMC, BMC Software, and the BMC Software logo are the exclusive properties of BMC Software, Inc., are registered with the U.S. Patent and Trademark Office, and may be registered or pending registration in other countries. All other BMC trademarks, service marks, and logos may be registered or pending registration in the U.S. or in other countries. All other trademarks or registered trademarks are the property of their respective owners.

CICS, IBM, IMS, MVS, z/Architecture, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

The information included in this documentation is the proprietary and confidential information of BMC Software, Inc., its affiliates, or licensors. Your use of this information is subject to the terms and conditions of the applicable End User License agreement for the product and to the proprietary and restricted rights notices included in the product documentation.

Restricted rights legend

U.S. Government Restricted Rights to Computer Software. UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure of any data and computer software by the U.S. Government is subject to restrictions, as applicable, set forth in FAR Section 52.227-14, DFARS 252.227-7013, DFARS 252.227-7014, DFARS 252.227-7015, and DFARS 252.227-7025, as amended from time to time. Contractor/Manufacturer is BMC SOFTWARE INC, 2101 CITYWEST BLVD, HOUSTON TX 77042-2827, USA. Any contract notices should be sent to this address.
Customer support

You can obtain technical support by using the BMC Support Central website or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, see “Before contacting BMC.”

Support website

You can obtain technical support from BMC 24 hours a day, 7 days a week at http://www.bmc.com/support. From this website, you can

- read overviews about support services and programs that BMC offers
- find the most current information about BMC products
- search a database for issues similar to yours and possible solutions
- order or download product documentation
- download products and maintenance
- report an issue or ask a question
- subscribe to receive proactive e-mail alerts when new product notices are released
- find worldwide BMC support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

Support by telephone or e-mail

In the United States and Canada, if you need technical support and do not have access to the web, call 1 800 537 1813 or send an e-mail message to customer_support@bmc.com. (In the subject line, enter SupID:<yourSupportContractID>, such as SupID:12345). Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC

Have the following information available so that Customer Support can begin working on your issue immediately:

- product information
  - product name
  - product version (release number)
  - license number and password (trial or permanent)

- operating system and environment information
  - machine type
  - operating system type, version, and service pack or other maintenance level such as PUT or PTF
  - system hardware configuration
  - serial numbers
  - related software (database, application, and communication) including type, version, and service pack or maintenance level

- sequence of events leading to the issue

- commands and options that you used

- messages received (and the time and date that you received them)
  - product error messages
  - messages from the operating system, such as file system full
  - messages from related software
License key and password information

If you have questions about your license key or password, use one of the following methods to get assistance:

- Send an e-mail message to customer_support@bmc.com.
# Contents

## About this book
- Related publications .................................................. 20
- Conventions .............................................................. 21
- Syntax statements .................................................... 21
- Syntax diagrams ....................................................... 22
- Summary of changes ................................................ 24

## Part 1  Introduction to using Fast Path Indexer/EP with PFX and IBM native indexes  

### Chapter 1  Introduction to Fast Path Indexer/EP  

- Overview .............................................................. 27
- Fast Path Indexer/EP terminology ................................ 28
- Fast Path Indexer/EP utilities for processing IBM native indexes .................................................. 29
- Fast Path Indexer/EP capabilities ................................ 29
- Advantages of using Fast Path Indexer/EP  
  - Enables creation of indexes ................................... 30
  - Improves integrity ................................................. 33
  - Enables selective duplication of source data ................. 33
  - Enables concurrent index rebuild during change or reload 33
  - Offers choice of index structure ................................ 34
  - Enables concurrent initialization of source and indexes 34
- Usage considerations ................................................ 34
  - System requirements ............................................. 34
  - Access method .................................................... 35
  - Authentication process ........................................ 35
  - Storage requirements ........................................... 35

### Chapter 2  Programming considerations for applications  

- Overview .............................................................. 37
- Fast Path Indexer/EP segments  
  - Primary DEDB segments ......................................... 38
  - Index segments ................................................... 39
- Relationship between segments .................................. 40
- Using an inverted hierarchy ........................................ 42
- Using status codes .................................................. 44
Using command codes ........................................... 45
Using Boolean operators ....................................... 46
Using subsequence fields with SHISAM or HISAM .......... 46
Performing key searches ....................................... 47
  Complete search ........................................... 48
  Partial search ............................................ 49
Search using a PCB without PROCSEQ ..................... 50
Search using a PCB with PROCSEQ ......................... 51
Search using a PCB with a stand-alone index ............... 52
Processing an index as a stand-alone database ............. 53

Chapter 3  Adding index functionality to the Fast Path environment 55

Part 2  Implementing PFX index databases 57

Chapter 4  Fast Path Indexer/EP PFX index macros 59
Overview .......................................................... 59
PFXCHILD macro ............................................... 60
  PFXCHILD keywords ...................................... 60
  PFXCHILD syntax diagrams ............................... 63
  PFXCHILD macro examples ............................... 63
PFXPART macro ................................................ 63
  PFXPART keyword ........................................ 64
  PFXPART syntax diagram ................................ 64
  PFXCHILD/PFXPART macro examples ................. 65
PFXXDFLD macro ............................................. 67
  PFXXDFLD keywords ..................................... 67
  PFXXDFLD syntax diagrams .............................. 69
  PFXXDFLD macro example ............................... 69
P FXGEN macro ................................................ 70
Creating multiple PFX index records from one source segment 70

Chapter 5  Fast Path Indexer/EP PFXAGEN utility 73
Overview .......................................................... 74
  Front-end process to ACBGEN ............................ 74
  PROCSEQ keyword ................................ ...... 76
  Shadow PCBs ............................................... 76
  PFXXLIB allocation and usage ........................... 77
Generating ACBs ............................................... 78
  DBD processing ........................................... 79
  PSB processing ............................................ 80
Executing PFXAGEN ......................................... 82
  JCL requirements ........................................ 82
  Control statement syntax ................................ 85
  Return codes ............................................... 86
  Report ....................................................... 87
Chapter 9  Index build utility 153

Introduction .......................................................... 154
Preparing for the index build ......................................... 154
Build function inputs and outputs ...................................... 155
DBRC considerations .................................................. 155
BUILD command keywords and subcommands ..................... 156
Selecting the database and areas ..................................... 157
Input to the build process .............................................. 158
  Allocating the DEDB area data set .................................. 158
  Allocating the XSCAN data set ...................................... 158
Allocating the secondary index databases ......................... 159
Specifying indexes to be built ...................................... 159
  Using the INDEX keyword or the &INDEX parameter .......... 160
Index partitioning ..................................................... 160
Performance considerations ......................................... 162
  Input caching .......................................................... 162
  Input thread processing ............................................. 162
  Index thread processing ............................................ 163
Customizing sort processing ......................................... 163
Sorting index information for the build function .................. 164
  Using the SORT keyword ............................................ 164
  Using the SORT_OPTION keyword ................................ 164
Rebuilding indexes during database change ......................... 165
Rebuilding indexes during database reload ......................... 166
Sample BUILD command scenarios ................................... 166
  Scenario 1: Build of PFX indexes defined to a DEDB by using XSCAN input .. 167
  Scenario 2: Build for specified indexes defined to a DEDB by using XSCAN input ............................................. 168
  Scenario 3: Build of specified indexes defined to a DEDB by using primary DEDB input ............................................. 169
  Scenario 4: Index build by using dynamic allocation of primary DEDB (INPUT_DSN_MASK) and indexes (OUTPUT_DSN_MASK) ............... 169
  Scenario 5: Build of IBM native indexes defined to a DEDB by using XSCAN input ............................................. 170

Chapter 10  Index validation utilities 173

Introduction .......................................................... 174
Verify function ......................................................... 174
Resynchronize function ............................................... 175
Verify function inputs and outputs ................................... 175
Resynchronize function inputs and outputs ......................... 176
DBRC considerations ................................................ 177
  Offline mode .......................................................... 177
  BMP mode ............................................................ 178
Index validation command language .................................. 178
Selecting the database and areas ..................................... 180
  Offline mode .......................................................... 180
  BMP mode ............................................................ 181
**Figures**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database structure for primary DEDB and secondary index databases</td>
<td>32</td>
</tr>
<tr>
<td>Root segment structure for a SHISAM index or HISAM index</td>
<td>39</td>
</tr>
<tr>
<td>Course Information primary DEDB (DIVSSD)</td>
<td>40</td>
</tr>
<tr>
<td>Course information—SHISAM index or HISAM index (DIVSI4)</td>
<td>41</td>
</tr>
<tr>
<td>PCB using an index</td>
<td>41</td>
</tr>
<tr>
<td>DL/I call qualified by the student name</td>
<td>41</td>
</tr>
<tr>
<td>DL/I get next call for a second occurrence</td>
<td>42</td>
</tr>
<tr>
<td>Primary DEDB with an inverted hierarchy</td>
<td>43</td>
</tr>
<tr>
<td>PCB for an inverted full-function database</td>
<td>43</td>
</tr>
<tr>
<td>PCB for an inverted primary DEDB</td>
<td>43</td>
</tr>
<tr>
<td>Segment search argument for an inverted database</td>
<td>44</td>
</tr>
<tr>
<td>Invalid SSA for an inverted hierarchy</td>
<td>44</td>
</tr>
<tr>
<td>Course information—primary DEDB (DIVSSD)</td>
<td>48</td>
</tr>
<tr>
<td>Course information—SHISAM index or HISAM index (DIVSI4)</td>
<td>48</td>
</tr>
<tr>
<td>PFXCHILD statement for a SHISAM or HISAM index</td>
<td>63</td>
</tr>
<tr>
<td>PFXCHILD macro example for a partitioned SHISAM index</td>
<td>63</td>
</tr>
<tr>
<td>PFXPART statement for defining partitioned PFX indexes</td>
<td>64</td>
</tr>
<tr>
<td>PFXCHILD/PFXPART macro example using the PFXPART macro to specify</td>
<td>65</td>
</tr>
<tr>
<td>additional partitioned PFX index databases</td>
<td>66</td>
</tr>
<tr>
<td>PFXCHILD/PFXPART macro example using the PFXPART macro to specify all</td>
<td>66</td>
</tr>
<tr>
<td>partitioned PFX index databases</td>
<td>69</td>
</tr>
<tr>
<td>PXXDFLD statement for SHISAM or HISAM index</td>
<td>69</td>
</tr>
<tr>
<td>PXXDFLD macro example for a SHISAM or HISAM index</td>
<td>69</td>
</tr>
<tr>
<td>Adding a PFXGEN macro to a primary DBD</td>
<td>70</td>
</tr>
<tr>
<td>Creating multiple PFX index records from the same source segment</td>
<td>70</td>
</tr>
<tr>
<td>PFX index registration process flow</td>
<td>75</td>
</tr>
<tr>
<td>PFXAGEN inputs and outputs</td>
<td>81</td>
</tr>
<tr>
<td>PFXAGEN sample JCL</td>
<td>82</td>
</tr>
<tr>
<td>Continuing keyword parameters on an additional line</td>
<td>85</td>
</tr>
<tr>
<td>PFXAGEN control statements and keywords</td>
<td>85</td>
</tr>
<tr>
<td>Fast Path Indexer/EP ACBGEN utility report</td>
<td>87</td>
</tr>
<tr>
<td>PFXAGEN sample JCL using dynamic allocation</td>
<td>89</td>
</tr>
<tr>
<td>DFSMDA macro sample</td>
<td>92</td>
</tr>
<tr>
<td>DFSUOCU0 inputs and outputs</td>
<td>93</td>
</tr>
<tr>
<td>DFSUOCU0 sample JCL</td>
<td>94</td>
</tr>
<tr>
<td>Modifying a primary DEDB to include Fast Path Indexer/EP macros</td>
<td>99</td>
</tr>
<tr>
<td>Recommended PFX index database source output by Fast Path Indexer/EP during primary DEDB DBDGEN</td>
<td>100</td>
</tr>
<tr>
<td>PCB example for a primary DEDB with PROCSEQ</td>
<td>102</td>
</tr>
<tr>
<td>PCB example for a primary DEDB without PROCSEQ</td>
<td>102</td>
</tr>
</tbody>
</table>
PCB example for a stand-alone index .............................................. 103
Structure for a standard PFX index format for user-written exit .......... 112
Structure for a compact PFX index format for user-written exit ........ 113
Partitioned exit routine processing modes .................................... 115
Partitioned exit routine example .................................................. 117
Basic Fast Path/EP Series product JCL for offline processing ........ 127
Specifying multiple function commands ....................................... 128
Identifying the DEDB area for an index build ............................... 128
Sample JCL to access the RECON data sets .................................. 132
BMP mode execution JCL ............................................................ 134
IFP mode execution JCL ............................................................. 137
Sample control statement for specifying all areas as input to XSCAN .... 139
Sample control statement for specifying selected areas as input to XSCAN . . 139
XSCAN function inputs and outputs ............................................. 142
Scanning all indexes to a primary DEDB ..................................... 146
Scanning an individual secondary index database to a primary DEDB ... 146
Scanning multiple secondary index databases to a primary DEDB by using &INDEX 146
Sorting index information by using SORT and SORT_OPTION keywords ... 149
Offline scan of all indexes defined to a DEDB ................................. 150
Offline scan for specified indexes defined to a DEDB ..................... 151
Online scan for specified indexes defined to a DEDB ....................... 152
Build function inputs and outputs .............................................. 155
Sample control statement for specifying all areas as input to the index build ... 157
Sample control statement for specifying selected areas as input to the index build ... 157
Building all indexes to a primary DEDB ..................................... 159
Building a single index to a DEDB .............................................. 160
Specifying multiple IX subcommands ......................................... 160
Building multiple secondary index databases to a primary DEDB by using &INDEX 160
PFXLIB view of partitioned SHISAM indexes and HISAM indexes ........ 161
Sorting index information by using SORT and SORT_OPTION keywords ... 165
Requesting index rebuild during DEDB restructuring ....................... 165
Requesting index threading during DEDB restructuring .................... 165
Requesting index rebuild during DEDB reload .............................. 166
Requesting index rebuild during DEDB reload .............................. 166
Build of PFX indexes defined to a DEDB by using XSCAN input ....... 167
Build for specified indexes defined to a DEDB using XSCAN input .... 168
Build of specified indexes defined to a DEDB by using primary DEDB input ... 169
Index build by using dynamic allocation of primary DEDB and indexes .... 170
Build of IBM native indexes defined to a DEDB by using XSCAN input .... 171
Verify function inputs and outputs ........................................... 176
Resynchronize function inputs and outputs .................................. 177
Sample control statement for specifying all areas as input to the index verify ... 181
Sample control statement for specifying selected areas as input to the index verify ... 181
Limiting areas to be processed for an index verify or resynchronize in BMP mode ... 181
Verifying all indexes for the primary DEDB
Resynchronizing all indexes for the primary DEDB
Verifying an individual index for the primary DEDB
Resynchronizing an individual index for the primary DEDB
Verifying multiple indexes for the primary DEDB
Resynchronizing multiple indexes for the primary DEDB
Specifying checkpoint call frequency during an online resync
Sorting index information by using SORT and SORT_OPTION keywords
Verify all indexes defined to a DEDB by using primary DEDB input
Verify specified indexes defined to a DEDB by using XSCAN input
Resynchronize all indexes defined to a DEDB by using primary DEDB input
Resynchronize specified indexes defined to a DEDB by using XSCAN input
PFXLIST inputs and outputs
PFXLIST sample JCL
PFXLIST control statements and keywords
Fast Path Indexer/EP Registration Verification report
Sample Fast Path databases for IMSID=IMS5
Example primary DEDB structure
Location code index (DIVSI1)
Instructor SSN index (DIVSI2)
Surname index (DIVSI3)
Area code index with subsequence fields (DIVSI4)
PRIMARY MENU
COURSE MENU
LOCATION MENU
INSTRUCTOR MENU
STUDENT MENU
PREREQUISITE MENU
ACCESS VIA STUDENT NAME panel
ACCESS VIA LOCATION CODE panel
ACCESS VIA INSTRUCTOR SSN panel
STUDENT NAME INDEX ONLY panel
STUDENT AREA CODE INDEX ONLY panel
Tables

Database and index terminology .................................................. 28
Fast Path Indexer/EP utilities for processing IBM native indexes .......... 29
Authorization tiers ....................................................................... 35
Fast Path Indexer/EP minimum storage requirements ......................... 36
Status codes ............................................................................... 45
Fast Path Indexer/EP usage process .............................................. 55
Fast Path Indexer/EP PFX index macros ......................................... 60
PFXCHILD keywords and parameters ........................................... 61
PFXPART keyword and parameter for SHISAM or HISAM indexes .......... 64
PFXXDFLD keywords and parameters .......................................... 67
Description of PFXAGEN control statements and keywords ............... 86
Maintenance functions ................................................................. 110
Registers at entry to user-written exit ......................................... 111
Parameter list for user-written exit .............................................. 112
Register 15 at time of return from user-written exit ......................... 112
Registers at entry to partitioned exit routine ................................ 117
Available operational modes by Fast Path Indexer/EP command .......... 126
IAREA parameters ...................................................................... 138
XSCAN command keywords and subcommands ............................. 143
DBRC settings for index build processing .................................. 156
BUILD command keywords and subcommands ............................. 156
DBRC settings for offline verify processing .................................. 178
VERIFY command keywords and subcommands ............................ 179
RESYNC command keywords and subcommands ........................... 179
PFXLIST control statements and keywords ................................ 198
Reason codes for RC=8 on message BMC110980E ......................... 206
Problem summary and documentation form .................................. 211
IMS transaction codes mapped to PF keys ................................... 225
About this book

This book discusses the Fast Path Indexer/EP product, which is an application enhancement tool that is included in the BMC Fast Path/EP Series of products. These products provide tools for database administrators, systems programmers, and technical support personnel involved in the management, maintenance, and performance enhancement of IBM® IMS™ Fast Path databases. This book is written primarily for IMS database administrators, and for other information technology professionals who use Fast Path Indexer/EP to create and maintain indexes to support Fast Path primary data-entry databases (DEDBs).

Some information in this book is intended to encourage interaction between the Fast Path database administrator and the applications programmer. This dialogue is instrumental to ensure success with Fast Path indexes, especially from an index maintenance perspective.

A working knowledge of IMS Fast Path and the IBM z/OS® operating system are essential to understanding the concepts and terminology that are described in this book.

This book provides information that is critical to preparing for index capability, and for leveraging secondary index databases in the Fast Path environment. This book contains many considerations for ensuring that Fast Path secondary index databases are designed and maintained in a manner that ensures compatibility with the primary databases, as well as correct integration with the applications they support.

Like most BMC documentation, this book is available in printed and online formats. To request printed books or to view online books and notices (such as release notes and technical bulletins), see the support website at http://www.bmc.com/support.

NOTE

Online books are formatted as PDF or HTML files. To view, print, or copy PDF books, use the free Adobe Reader from Adobe Systems. If your product installation does not install the reader, you can obtain the reader at http://www.adobe.com.

The software also offers online Help. To access Help, press F1 within any product or click the Help button in graphical user interfaces (GUIs).
## Related publications

The following related publications supplement this book and the online Help:

<table>
<thead>
<tr>
<th>Category</th>
<th>Document</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>installation</td>
<td><strong>Installation System User Guide</strong></td>
<td>provides installation procedures and optional post-installation procedures for database administrators and technical support personnel involved with the initial installation and maintenance installation of Fast Path Indexer/EP and other BMC Fast Path products for IMS</td>
</tr>
<tr>
<td>product use</td>
<td><strong>Fast Path/EP Series Reference Manual</strong></td>
<td>provides reference information about the following topics that are pertinent to Fast Path Indexer/EP:</td>
</tr>
</tbody>
</table>
|                     |                                               |   • JCL conventions  
   • conventions for using the commands, subcommands, and keywords that control product functionality                                                                                                    |
| notices             | **release notes, flashes, technical bulletins** | provides updates to the installation instructions and last-minute product information                                                                                                                       |

From the BMC Support Central website ([http://www.bmc.com/support](http://www.bmc.com/support)), you can

- download a zipped set of documentation PDFs from each product’s EPD page

- link to the BMC Documentation Center ([https://webapps.bmc.com/infocenter/index.jsp](https://webapps.bmc.com/infocenter/index.jsp)) to browse documentation sets, or to view video demos (short overviews of selected product concepts, tasks, or features)

- view individual product documents (books and notices) within the “A – Z Supported Product List”

You can order hardcopy documentation from your BMC sales representative or from the support site. You can also subscribe to proactive alerts to receive e-mail messages when notices are issued.
Conventions

This book uses the following special conventions:

- All syntax, operating system terms, and literal examples are presented in this typeface.

- Variable text in path names, system messages, or syntax is displayed in italic text:
  
  testsys/instance/file

- The symbol => connects items in a menu sequence. For example, Actions => Create Test instructs you to choose the Create Test command from the Actions menu.

- Revision bars in the document mark changes that clarify or correct existing information or that provide new information. Revision bars do not mark editorial changes, formatting changes, or corrections of typographical errors unless these updates significantly affect your use of the information.

Syntax statements

The following example shows a sample syntax statement:

```
COMMAND KEYWORD1 [KEYWORD2 | KEYWORD3] KEYWORD4={YES | NO} fileName...
```

The following table explains conventions for syntax statements and provides examples:

<table>
<thead>
<tr>
<th>Item</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items in italic type represent variables that you must replace with a</td>
<td>alias</td>
</tr>
<tr>
<td>name or value. If a variable is represented by two or more words,</td>
<td>databaseDirectory</td>
</tr>
<tr>
<td>initial capitals distinguish the second and subsequent words.</td>
<td>serverHostName</td>
</tr>
<tr>
<td>Brackets indicate a group of optional items. Do not type the</td>
<td>[tableName, columnName, field]</td>
</tr>
<tr>
<td>brackets when you enter the option. A comma means that you can</td>
<td>[-full, -incremental, -level]</td>
</tr>
<tr>
<td>choose one or more of the listed options. You must use a comma to</td>
<td>(UNIX)</td>
</tr>
<tr>
<td>separate the options if you choose more than one option.</td>
<td></td>
</tr>
</tbody>
</table>
### Syntax diagrams

The following figure shows the standard format for syntax diagrams:

- **Item**
  - Braces indicate that at least one of the enclosed items is required. Do not type the braces when you enter the item.
  - Example: `{DBDName | tableName}
  
  **UNLOAD**
  
  device={disk | tape,
  fileName | deviceName}

- **Item**
  - A vertical bar means that you can choose only one of the listed items. In the example, you would choose either **commit** or **cancel**.
  - Example: `{commit | cancel}
  
  **(UNIX)**

- **Item**
  - An ellipsis indicates that you can repeat the previous item or items as many times as necessary.
  - Example: `columnName . . .`

---

<table>
<thead>
<tr>
<th>Item</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Braces indicate that at least one of the enclosed items is required. Do not type the braces when you enter the item.</td>
<td>`{DBDName</td>
</tr>
<tr>
<td><strong>UNLOAD</strong></td>
<td></td>
</tr>
<tr>
<td>device={disk</td>
<td>tape, fileName</td>
</tr>
<tr>
<td>{a</td>
<td>c} (UNIX)</td>
</tr>
<tr>
<td>A vertical bar means that you can choose only one of the listed items. In the example, you would choose either <strong>commit</strong> or <strong>cancel</strong>.</td>
<td>`{commit</td>
</tr>
<tr>
<td><strong>(UNIX)</strong></td>
<td></td>
</tr>
<tr>
<td>An ellipsis indicates that you can repeat the previous item or items as many times as necessary.</td>
<td><code>columnName . . .</code></td>
</tr>
</tbody>
</table>
The following example illustrates the syntax for a DELETE statement. Because the FROM keyword, alias variable, and WHERE clause are optional, they appear below the main command line. In contrast, the tableName variable appears on the command line because the table name is required. If the statement includes a WHERE clause, the clause must contain a search condition or a CURRENT OF clause. (The searchCondition variable appears on the main line for the WHERE clause, indicating that this choice is required.)

DELETE
tableName
FROM
alias

WHERE
searchCondition
cursorName

The following guidelines provide additional information about syntax diagrams:

- Read diagrams from left to right and from top to bottom.

- A recursive (left-pointing) arrow above a stack indicates that you may choose more than one item in the stack.

- An underlined item is a default option.

- If a diagram shows punctuation marks, parentheses, or similar symbols, you must enter them as part of the syntax. Asterisks are exceptions. An asterisk in a diagram indicates a reference note.

- In general, IBM MVS™ commands, keywords, clauses, and data types are displayed in uppercase letters. However, if an item can be shortened, the minimum portion of the MVS command or keyword might be displayed in uppercase letters with the remainder of the word in lowercase letters (for example, CANcel).

- The following conventions apply to variables in syntax diagrams:

  — Variables typically are displayed in lowercase letters and are always italicized.

  — If a variable is represented by two or more words, initial capitals distinguish the second and subsequent words (for example, databaseName).
Summary of changes

For detailed information about enhancements, changes, and corrections that are included in your version of the product, see the product release notes. The release notes are available from the BMC Support Central page (http://www.bmc.com/support).
This part presents the following topics:

Chapter 1
Introduction to Fast Path Indexer/EP .............................................. 27

Chapter 2
Programming considerations for applications .............................. 37

Chapter 3
Adding index functionality to the Fast Path environment .............. 55
Introduction to Fast Path Indexer/EP

This chapter provides an overview of Fast Path Indexer/EP (PFX), a product that lets you build indexes for Fast Path primary data-entry databases (DEDBs). Read this chapter if you are not familiar with creating and using indexes with hierarchically structured databases, or if you are a new user of Fast Path Indexer/EP.

This chapter discusses the following topics:

Overview ................................................................. 27
Fast Path Indexer/EP terminology .................................... 28
Fast Path Indexer/EP utilities for processing IBM native indexes ........................................ 29
Fast Path Indexer/EP capabilities ...................................... 29
Advantages of using Fast Path Indexer/EP ......................... 30
  Enables creation of indexes ........................................ 31
  Improves integrity .................................................... 33
  Enables selective duplication of source data ........................ 33
  Enables concurrent index rebuild during change or reload ........... 33
  Offers choice of index structure .................................... 34
  Enables concurrent initialization of source and indexes ............... 34
Usage considerations .................................................. 34
  System requirements ............................................... 34
  Access method ....................................................... 35
  Authentication process .............................................. 35
  Storage requirements ............................................... 35

Overview

The BMC Fast Path Indexer/EP product is an application enhancement tool that is included in the BMC Fast Path/EP Series of products. The Fast Path/EP Series consists of the following products:

- Fast Path Analyzer/EP
- Fast Path Indexer/EP
These products deliver a range of functions for reorganizing databases, managing spaces, and associating indexes with Fast Path databases.

## Fast Path Indexer/EP terminology

Table 1 defines basic terms that are key to understanding Fast Path.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>primary data-entry database (DEDB)</td>
<td>a direct-access database that consists of one or more areas, with each area containing both root segments and dependent segments. The primary DEDB was formerly called the existing database, existing DEDB, or source database.</td>
</tr>
<tr>
<td>secondary index</td>
<td>a PFX index or IBM native index that provides access to a primary DEDB by a path other than the path that the database definition (DBD) provides.</td>
</tr>
<tr>
<td>secondary index database</td>
<td>a secondary index that is processed as an independent database (formerly called an index database).</td>
</tr>
<tr>
<td>PFX index</td>
<td>a secondary index database that is defined in the DBD for the primary DEDB. The secondary index is defined by the Fast Path Indexer/EP PFXCHILD and PFXXDFLD macro statements. Fast Path Indexer/EP uses other macro statements to specify partitioned index DBD names and build the definition block. For more information, see Chapter 4, “Fast Path Indexer/EP PFX index macros.”</td>
</tr>
<tr>
<td>IBM native index</td>
<td>(IMS version 12.1 and later) a secondary index database that is defined in the DBD for the primary DEDB. The IBM LCHILD or XDFLD statement defines the secondary index. For more information, see the IBM documentation.</td>
</tr>
</tbody>
</table>
Fast Path Indexer/EP utilities for processing IBM native indexes

Fast Path Indexer/EP utilities support both PFX indexes and IBM native indexes for DEDB databases.

For IBM natives indexes, use Table 2 for guidance to locate topics for using Fast Path Indexer/EP utilities to process IBM native indexes.

Table 2  Fast Path Indexer/EP utilities for processing IBM native indexes

<table>
<thead>
<tr>
<th>Topic</th>
<th>Location in this book</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage considerations</td>
<td>Chapter 1, “Introduction to Fast Path Indexer/EP”</td>
</tr>
<tr>
<td>Index utility commands and processing modes</td>
<td>Chapter 7, “Index utility commands and processing modes”</td>
</tr>
<tr>
<td>Index scan utility (XSCAN command)</td>
<td>Chapter 8, “Index scan utility”</td>
</tr>
<tr>
<td>Index build utility (BUILD command)</td>
<td>Chapter 9, “Index build utility”</td>
</tr>
<tr>
<td>Index validation utilities (VERIFY command)</td>
<td>Chapter 10, “Index validation utilities”</td>
</tr>
</tbody>
</table>

a The RESYNC command can process only PFX indexes. It cannot process IBM native indexes.

Fast Path Indexer/EP capabilities

Fast Path Indexer/EP lets you build and maintain indexes that are associated with primary DEDBs. When IMS updates the database, Fast Path Indexer/EP updates the associated indexes. The product can also ensure that the database and its indexes remain synchronized if an IMS failure or database recovery occurs.

Fast Path Indexer/EP lets your applications perform the following functions:

- partition an index

  You can build a partitioned index and include as many as 500 partitions in the index. Each partition can be up to 4 GB.

- process an index as a stand-alone database

  You can process an index that was created with Fast Path Indexer/EP as a stand-alone database for retrievals. This processing method lets you retrieve information while improving system performance.
Advantages of using Fast Path Indexer/EP

- create 500 indexes

  Whether you choose the SHISAM structure or the HISAM structure, you can build up to 500 indexes. If you partition an index, you can include 500 partitions.

- create index keys from five source fields

  You can use up to five fields from the primary DEDB segment to create the index key. You can use any field in the segment as the key in the index. If the primary DEDB data is not unique, you must use the subsequence field to force uniqueness.

- create multiple index records

  You can create multiple index records from one source segment of an index.

- store duplicate data

  You can include up to five duplicate data fields that you obtained from the primary DEDB segment in the index.

- maintain indexes

  Fast Path Indexer/EP lets you populate the index only with data that meets specific criteria. For example, you can ignore records from the primary DEDB that contain blanks or zeros by using the NULLVAL keyword or the sparse indexing exit (secondary index database maintenance exit) routine.

- use an inverted hierarchy

  Fast Path Indexer/EP supports an inversion of hierarchy up to the root segment. To use an alternate processing sequence, you must specify the PROCSEQ keyword.

Advantages of using Fast Path Indexer/EP

Fast Path Indexer/EP provides the following advantages:

- Fast Path Indexer/EP provides index building and maintenance capabilities for DEDBs.

- In a manner similar to how secondary indexes work with IMS full-function databases, an index to a Fast Path DEDB lets your applications process the primary DEDB in an alternate processing sequence.
Because the index contains only a portion of the data in the primary DEDB, processing time is reduced for scan or browse operations.

Using an index increases transaction throughput.

Fast Path Indexer/EP uses IMS services to update the index automatically when changes are made to the primary DEDB.

You can use standard IMS services to recover an index to the same point in time as the primary DEDB.

You can rebuild an index from the recovered primary DEDB by using the product’s batch utilities.

**Enables creation of indexes**

In efficient database designs, segments and fields are arranged to best fit the requirements of the most frequently used transactions. Even the most well-designed database, however, can run more efficiently if it enables applications to use an alternate processing sequence when needed. A secondary index database solves this dilemma; you can use a second (alternate) processing sequence for the same database without disrupting the primary processing sequence that is defined in the primary DEDB.

Figure 1 shows an example of a primary DEDB and an associated secondary index database. The primary DEDB accesses course data through the root segment (Course).
For an application to access this primary DEDB, the application must know the course number (root segment) before accessing any other information. If you use an index, however, the entry key into the database can be any segment; it does not have to be the root segment.

For example, assume that your application program could benefit by accessing the database through the Location segment, not the root segment (Course). Using a secondary index database, the application program accesses the database in this alternate manner. The application program accesses the database directly at the Location segment by using the Location code as the key. The program can then access all dependent segments of Location.

Path calls enable the application program to retrieve any segment in the path from the source segment (Location) to the root segment (Course). After the application accesses the Location segment, the segment is placed into the application’s I/O area. This type of access speeds processing time by reducing the number of times that the application must access the primary DEDB.
Improves integrity

Fast Path Indexer/EP addresses the following integrity issues:

- primary DEDB maintenance

  Fast Path Indexer/EP does not alter or interfere with the primary DEDB.

- index maintenance

  Fast Path Indexer/EP uses IMS services to provide index maintenance when changes are made to the primary DEDB. To ensure integrity, Fast Path Indexer/EP prohibits the application from updating the index directly.

- recovery

  Fast Path Indexer/EP lets you register an index with the IMS Database Recovery Control (DBRC) feature. If you need to recover the primary DEDB, you must perform a synchronous recovery of the index and the primary DEDB. You can recover the index to the same point in time as the primary DEDB, or you can rebuild the index from the recovered primary DEDB by executing the Fast Path Indexer/EP BUILD command.

Enables selective duplication of source data

Creating an index by using Fast Path Indexer/EP is an efficient way to duplicate data from a primary DEDB. If you process your index as a stand-alone database, you can create commands that duplicate only specific data from the primary DEDB. This capability lets you process only the index for that data. Otherwise, you would have to process the entire primary DEDB, which would be time-consuming and costly.

Enables concurrent index rebuild during change or reload

If you change the primary DEDB, you can simultaneously rebuild the associated secondary index database via the offline CHANGE or RELOAD commands; these commands are available in the Fast Path Reorg/EP product.
Offers choice of index structure

When defining Fast Path Indexer/EP indexes, you can use a SHISAM structure or HISAM structure. Both of these database structures store the index data in sequential order.

Consider the following items based on index activity:

- SHISAM does not reuse the space for deleted segments.
- HISAM reuses space from deleted segments. HISAM is a better choice for high-activity indexes.

Enables concurrent initialization of source and indexes

If you are initializing a DEDB that has registered indexes of the DEDB structure type, Fast Path Indexer/EP lets you simultaneously initialize the primary DEDB and its indexes in one job step. You can initialize the primary DEDB and indexes when executing the INITIALIZE command that is available in the Fast Path Reorg/EP product.

Usage considerations

This section lists system, access, authentication, and storage considerations for Fast Path Indexer/EP.

System requirements

- IBM z/Architecture® mainframes
  
  BMC licenses Fast Path Indexer/EP to run on specific CPUs.

- IBM z/OS 1.10 or later

- supported releases of IMS

- APF authorization for the STEPLIB containing the Fast Path Indexer/EP load library

- standard IMS RESLIB (SDFSRESL) for execution
Access method

When deciding upon a particular type of access method, consider the following information:

- You must code the primary DEDB with the PROCSEQ keyword (for PFX indexes) or PROCSEQD keyword (for IBM native indexes) in the PSB before you can access the data in the primary DEDB through an alternate processing sequence method.

- To access the index as a stand-alone database, you must provide a PCB for that index. You should specify a read-only PROCOPT.

Authentication process

When processing indexes, the Fast Path Indexer/EP product uses the tiered password feature that is available within the security system for the product. The tiered passwords allow the product’s XSCAN, BUILD, VERIFY, and RESYNC commands to process PFX indexes, IBM native indexes, or both, as shown in Table 3.

Table 3  Authorization tiers

<table>
<thead>
<tr>
<th>Tier</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>enables the XSCAN, BUILD, VERIFY, and RESYNC commands to process only PFX indexes</td>
</tr>
<tr>
<td>1</td>
<td>enables the XSCAN, BUILD, and VERIFY commands to process only IBM native indexes</td>
</tr>
<tr>
<td>2</td>
<td>reserved for future use</td>
</tr>
<tr>
<td>3</td>
<td>enables the XSCAN, BUILD, and VERIFY commands to process PFX and IBM native indexes</td>
</tr>
</tbody>
</table>

Tier 3 also enables the RESYNC command to process PFX indexes. The commands to process the indexes might be executed in different job steps.

Storage requirements

Fast Path Indexer/EP obtains storage from the extended common service area (ECSA) when possible, regardless of the options that are defined in the IMS system. Fast Path Indexer/EP must include the minimum storage requirements shown in Table 4.
<table>
<thead>
<tr>
<th>Use</th>
<th>Area</th>
<th>Amount</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>system initialization</td>
<td>modules</td>
<td>10 KB</td>
<td>CSA</td>
</tr>
<tr>
<td>normal execution</td>
<td>modules</td>
<td>63 KB</td>
<td>ECSA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 KB</td>
<td>CSA</td>
</tr>
<tr>
<td>control blocks</td>
<td></td>
<td>5 KB</td>
<td>CSA</td>
</tr>
<tr>
<td>per PST</td>
<td></td>
<td>1.1 KB</td>
<td>ECSA</td>
</tr>
<tr>
<td>additional storage</td>
<td>definition blocks</td>
<td>84 bytes per secondary index database</td>
<td>ECSA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>212 bytes per primary DEDB</td>
<td>ECSA</td>
</tr>
</tbody>
</table>
Programming considerations for applications

This chapter discusses application programming issues that you should consider before using your applications with Fast Path Indexer/EP.

This chapter discusses the following topics:

Overview .......................................................... 37
Fast Path Indexer/EP segments .................................. 38
  Primary DEEB segments ...................................... 38
  Index segments ............................................... 39
  Relationship between segments ........................... 40
Using an inverted hierarchy .................................... 42
Using status codes .............................................. 44
Using command codes ......................................... 45
Using Boolean operators ....................................... 46
Using subsequence fields with SHISAM or HISAM. ......... 46
Performing key searches ....................................... 47
  Complete search ............................................. 48
  Partial search ............................................... 49
  Search using a PCB without PROCSEQ ................. 50
  Search using a PCB with PROCSEQ .................. 51
  Search using a PCB with a stand-alone index .......... 52
Processing an index as a stand-alone database .............. 53

Overview

Fast Path Indexer/EP supports the following functions:

- inverted hierarchy for retrieval
- IMS status codes
Fast Path Indexer/EP segments

In IMS, a segment is a unit of access to a database. A segment is the smallest amount of data that a DL/I operation can transfer. Fast Path Indexer/EP uses several segment types to keep track of indexed data. This section discusses the types of primary DEDB and index segments that are used with the product.

Primary DEDB segments

The primary DEDB contains target and source segments:

- The target segment is the segment on the primary DEDB that your application program needs to retrieve. The concatenated key field in the pointer segment (which resides on the index) points to the target segment.

- The source segment resides on the primary DEDB and contains the fields that the pointer segment maintains as its key fields. Fast Path Indexer/EP copies this data from the source segment and places it in the corresponding key fields of the pointer segment in the index.

The source and target segments can be the same, or the source can be a dependent of the target; the source cannot be a parent of the target. All segments residing in the hierarchical path of a primary DEDB, from the root segment to the source segment, can be keyed or non-keyed.
A SHISAM or HISAM index contains only a root segment. The root segment contains the sequence field of the index and the symbolic pointer. The root segment contains the same data as the DEDB index root and pointer segments. Because the SHISAM or HISAM index root segment is fixed length, it does not contain a length field.

The key field contains the fields from the primary DEDB that are to be indexed. These fields comprise the sequence field of the index. Fast Path Indexer/EP places them in the SHISAM or HISAM index root segment in the order that they were defined during the index registration process. You can include up to five fields from the primary DEDB.

The SHISAM or HISAM structure also includes optional subsequence fields. These fields are added to the index segment key data to make the pointer segment key unique. For more information about how to use subsequence fields, see “Using subsequence fields with SHISAM or HISAM” on page 46.

The symbolic pointer, which is the concatenated key to the target segment, refers to the segment that your application program is retrieving. Figure 2 illustrates the structure of a SHISAM or HISAM index root segment.

**Figure 2 Root segment structure for a SHISAM index or HISAM index**

<table>
<thead>
<tr>
<th>key</th>
<th>symbolic pointer</th>
<th>duplicate data</th>
<th>control data</th>
</tr>
</thead>
<tbody>
<tr>
<td>search field&lt;br&gt;The length of this field is dependent on the length and number of fields from the primary DEDB.</td>
<td>subsequence fields (optional)</td>
<td>concatenated key (varies)</td>
<td>up to 5 fields</td>
</tr>
</tbody>
</table>

If Fast Path Indexer/EP performs calls to SHISAM or HISAM indexes with subsequence fields and returns an AU status code, the SSASIZE keyword must be added or the SSASIZE value must be increased in the PSBGEN statements.
Relationship between segments

Figure 3 illustrates how the root segment (Course) of the primary DEDB contains the course number and course name. It also illustrates how the root segment is keyed by the course number, how the Location segment is keyed by the Location code, and how the Location segment has dependent segments. These dependent segments describe the instructors (Instructor segment) and the students (Student segment). This path describes which instructors and which students are associated with what course at a particular location.

Figure 3 Course Information primary DEDB (DIVSSD)

Figure 4 illustrates how you could design a SHISAM or HISAM index to be associated with the Course Information primary DEDB in Figure 3.
Assume that you want to access the Course Information primary DEDB through the student name, and you need a list of all courses that each student is taking. Using Figure 3 as an example, the Student segment becomes the source segment because this segment contains student names. This Student segment is also the key field of the index, as illustrated in Figure 4. The Course segment is the target segment because the Course segment contains the course names that your application program needs to retrieve.

Because your application program needs to retrieve data through a student name, you create an index that includes the student name as the key of the root segment. This lets you maintain an index that sequences through the student name.

You must also define a PSB, which contains a PCB indicating to Fast Path Indexer/EP that your application program needs to access the Course Information primary DEDB using this index. This index includes the PROCSEQ keyword, indicating an alternate processing sequence. Figure 5 shows an example of this alternate processing sequence PCB.

Suppose that you have coded your application program to issue a DL/I call that qualifies only the student name (SMITH). Figure 6 shows an example of this type of call.
Assume that only two courses are associated with this student name. If you issue the call shown in Figure 6, the first occurrence of a course for the student name of SMITH will be returned in the I/O area (DB-IOAREA) as follows:

```
'C0001ECON101'
```

The first five bytes of the I/O area show the course number, followed by seven bytes that indicate the course name.

To retrieve the next occurrence of the course for SMITH, you must issue a GET NEXT (GN) call as shown in Figure 7 on page 42.

**Figure 7  DL/I get next call for a second occurrence**

```
CALL 'CBLTDLI' USING GN-FUNC, DIVSSD-PCB, DB-IOAREA, SSA-DIVSSD-COURSE.
```

This call will be returned in the I/O area as follows:

```
'C0002CALC102'
```

If you issue the GN call again for this example, Fast Path Indexer/EP issues a GE status code to indicate that no more courses are associated with this student. You would then have a complete course list for the student named SMITH.

---

**Using an inverted hierarchy**

Fast Path Indexer/EP supports an inversion of hierarchy up to the root segment for retrieval processing. Fast Path Indexer/EP does not physically invert the hierarchy for an alternate processing sequence method. It does not add extra pointers to the primary DEDB or any associated indexes. To indicate to Fast Path Indexer/EP that you want to use an alternate processing sequence, you must use the PROCSEQ keyword. A retrieval can proceed *no further* than the root.

The inversion of hierarchy feature is not unique to Fast Path Indexer/EP; IMS also inverts the hierarchy when you include the PROCSEQ keyword on the PCB. However, IMS does not view the PROCSEQ keyword located on the primary DEDB as the true PROCSEQ and does not invert the SENSEG statements that follow the PCB. You must code the SENSEG statements in the original processing sequence order.
When you indicate an alternate processing sequence by using an unqualified GET NEXT (GN) call, Fast Path Indexer/EP begins at the target segment and processes in reverse order (from the left) until it reaches the root segment. Figure 8 illustrates an inverted example of the primary DEDB.

**Figure 8   Primary DEDB with an inverted hierarchy**

![Diagram of an inverted hierarchy]

When using an inverted hierarchy, Fast Path Indexer/EP works in a way similar to the way that IMS works: it begins at the Instructor segment, proceeds to the Location segment, then proceeds to the Course segment. Fast Path Indexer/EP proceeds along a hierarchical path of the target segment only, up to the root segment when you issue an unqualified GN call.

The PSB shown in Figure 9 illustrates a PCB for a full-function database that is inverted:

**Figure 9   PCB for an inverted full-function database**

```plaintext
PCB  TYPE=DB,PROCSEQ=DIVSI2,DBDNAME=FULLFUNC,
     KEYLEN=24,PROCOPT=A
SENSEG NAME=INSTR,PARENT=0
SENSEG NAME=LOCN,PARENT=INSTR
SENSEG NAME=COURSE,PARENT=LOCN
SENSEG NAME=STUDENT,PARENT=LOCN
```

The PSB shown in Figure 10 illustrates a PCB for a primary DEDB that will be inverted by Fast Path Indexer/EP:

**Figure 10   PCB for an inverted primary DEDB (part 1 of 2)**

```plaintext
PCB  TYPE=DB,PROCSEQ=DIVSI2,DBDNAME=DIVSSD,
     KEYLEN=24,PROCOPT=A
SENSEG NAME=COURSE,PARENT=0
```
The segment search argument (SSA) shown in Figure 11 could be used with a full-function database or a Fast Path Indexer/EP database:

Using the inverted hierarchy illustrated in Figure 8, your application program cannot access the Prerequisite segment because this segment is located beyond the original root segment. Therefore, the SSA shown in Figure 12 is invalid. The call would return an AC status code.

Like an unqualified GET UNIQUE (GU) call, an unqualified GN call cannot proceed beyond the root segment. An unqualified GN call will retrieve only those segments that are in a direct line from the target segment to the root segment.

Using the inverted hierarchy illustrated in Figure 8, if you issued another unqualified GN call, the next Instructor segment would be retrieved.

Using status codes

Fast Path Indexer/EP and IMS use the same status codes. Fast Path Indexer/EP recognizes a status code as a two-character code that resides in the PCB mask. The status code indicates the outcome of a DL/I call.

The status codes in Table 5 relate exclusively to indexes. For more information about these codes, see the IBM IMS/ESA for Application Programming manual.
Using command codes

Fast Path Indexer/EP supports most IMS command codes.

**NOTE**

Fast Path Indexer/EP *does not* support the IMS command code C related to concatenated keys.

The IMS command code D is the only IMS command code that is associated with indexes. Command code D instructs the application program to obtain all segments in a particular path and place them in the I/O area. For example, if you have a database that includes information about a university and you want to obtain the Instructor, Location, and Course segments in a certain path, use the following call:

```
GU  INSTR  *D--(SRCVALU   EQ000007000)
    LOCN   *D--(LOCNCD    EQDDDD)
    COURSE *D--(COURSE#   EQ00005)
```

To obtain only the Instructor and Course segments, use the following call:

```
GU  INSTR  *D--(SRCVALU   EQ000007000)
    LOCN   *---(LOCNCD    EQDDDD)
    COURSE *D--(COURSE#   EQ00005)
```

If you do not specify the last segment in a path (in this example, Course), IMS always assumes that you want to obtain it.

**NOTE**

Fast Path Indexer/EP does not support path calls that insert, replace, or delete through a PROCSEQ PCB.

---

**Table 5  Status codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE</td>
<td>During a retrieval, Fast Path Indexer/EP could not locate a segment on the primary DEDB, despite that same segment being located on the associated index. Use the BUILD command to rebuild the index. Then, use the VERIFY command to verify the information about the index.</td>
</tr>
<tr>
<td>NI</td>
<td>You have tried to insert a duplicate segment in an index that you defined as unique. Fast Path Indexer/EP initiates an internal ROLB call to IMS to back out database updates created since the last commit point.</td>
</tr>
</tbody>
</table>
Using Boolean operators

Fast Path Indexer/EP allows Boolean operators to access data through an index, with one exception. If you issue a qualified call against an index that is using the PROCSEQ keyword and Boolean operators, you cannot mix the XDFLD name and the target segment sequence field name. For example, if you want to index on a Location segment in an index, the following SSA is invalid:

```
GU LOCN *-(XDFLD EQHOUS*LOCNCD NEGORG)
```

If you used this SSA, IMS would issue an AJ status code.

Using subsequence fields with SHISAM or HISAM

You can use a subsequence field to force a SHISAM or HISAM key to be unique; however, you should not use the subsequence field as part of your SSA key value.

The subsequence field is internal to IMS. Fast Path Indexer/EP automatically converts your SSA value to a Boolean search call so that you can retrieve each occurrence of this key value by issuing a GN call.

When Fast Path Indexer/EP issues calls to SHISAM or HISAM indexes with subsequence fields and Fast Path Indexer/EP returns an AU status code, the SSASIZE keyword must be added or increased in the PSBGEN statement. The SSASIZE keyword specifies the maximum total length of all SSAs that the application program uses.

ACBGEN allows 280 bytes of SSA storage for each hierarchical level in the PCB. A one-level SHISAM or HISAM index will get only 280 bytes for the SSA, but Fast Path Indexer/EP (depending on the key size) might require more. Fast Path Indexer/EP returns the AU status code if its internal call is larger than the SSA storage that the PSBGEN specifies.

The size specification determines the amount of main storage that is reserved in the PSB work pool to hold a copy of the user’s SSA strings during the execution of this application program. If this value is not specified in the PSBGEN statement, the ACB utility program calculates a maximum SSA size to be used as a default. The calculated size is the maximum number of hierarchical levels in any PCB within this PSB, multiplied by 280. The value specified is in bytes, with a maximum of 256,000 bytes.
Performing key searches

Fast Path Indexer/EP lets you perform complete and partial key searches. This section discusses the following topics:

- complete searches
- partial searches
- searches using a PCB that does not include the PROCSEQ keyword (primary sequence processing method)
- searches using a PCB that includes the PROCSEQ keyword (alternate sequence processing method)
- searches using a PCB with a secondary index

**NOTE**

Fast Path Indexer/EP tries to keep calls to the primary DEDB and associated indexes to a minimum by restricting the use of non-keyed search fields. All field names that you include in an SSA must be the same as the XDFLD name or the sequence field name. If you use any other field name, IMS issues an AK status code.

**NOTE**

When IMS is running under IBM CICS® with DBCTL, the PSB work pool requirement cannot exceed 64 KB. The major components of this pool requirement are IOASIZE and SSASIZE. When the PSB is built into ACBLIB, ACBGEN message DFS05891 indicates the total work pool space requirement of the PSB.

*Figure 13* shows a primary DEDB (DIVSSD), and *Figure 14* shows a SHISAM or HISAM index (DIVSI4). The Student segment is the target segment and source segment.
When your application program performs a key search, it usually performs a complete search because the application program usually knows the entire contents of the key upon which the search is based.

Assume that your application program tries to access the Student segment on DIVSSD by using DIVSI4. The application program performs a search using the Student Area Code (STUDAREA) field, which is three bytes long. The program knows the entire value of the STUDAREA field.
If your program wants to retrieve data from Area Code 888, it issues the following DL/I call:

```cbltdli
CALL 'CBLTDLI' USING FUNC-GU
DIVSSD-PCB
USER-IO-AREA
SSA-DIVSSD-QUAL
```

The qualified SSA in the program would look like the following example:

```sql
STUDENT *-(SRCVALU =888)
```

In the SSA, STUDENT represents the DEDB target segment to be indexed and SRCVALU represents the field name in the PFXXDFLD macro.

When your application program successfully retrieves the Student segment, the following information is returned to the program:

- **I/O area:** STUDENT SEGMENT
- **Key feedback area:** 888

**Partial search**

If your application program knows only a portion of the search key, it can perform a partial search. IMS requires that your application program pad the entire length of the search key. If the program does not, IMS issues an AJ status code.

Using the same examples shown in Figure 13 on page 48 and Figure 14 on page 48, assume that your application program wants to perform a search but knows only the first two bytes of the three-byte STUDAREA field (the search field).

If your program wants to retrieve data from Area Code 888, it issues the following DL/I call:

```cbltdli
CALL 'CBLTDLI' USING FUNC-GU
DIVSSD-PCB
USER-IO-AREA
SSA-DIVSSD-QUAL
```

The qualified SSA in the program would look like the following example:

```sql
STUDENT *-(SRCVALU GE880)
```
The lowercase ‘o’ in ‘88o’ represents low values, assuming that no other area code is found between 88o and 888. When your application program successfully retrieves the Student segment, the following information is returned to the program:

I/O area: STUDENT SEGMENT
Key feedback area: 888

**Search using a PCB without PROCSEQ**

Assume that your application program issues a DL/I call referring to a PCB that does not contain the PROCSEQ keyword. This type of call indicates to Fast Path Indexer/EP that the application program is not using the index to access the primary DEDB. Code the PCB as follows:

```plaintext
PCB DBDNAME=DIVSSD,KEYLEN=99,PROCLOPT=A
```

The application program tries to retrieve all Student segments with number **11111**. It issues the following DL/I call:

```plaintext
CALL 'CBLTDLI' USING FUNC-GU
   DIVSSD-PCB
   USER-IO-AREA
   SSA-DIVSSD-QUAL
```

The qualified SSA in the program would appear as follows:

```plaintext
COURSE   *-D
LOCN     *-D
STUDENT  *-(STUD# =11111)
```

When your application program successfully retrieves the Student segment, the following information is returned to the program:

I/O area: COURSE, LOCATION and STUDENT SEGMENTS
Key feedback area :COURS# LOC# STUD#

Because you specified the D command code, IMS returns the Course and Location segments in the I/O area. See “Using command codes” on page 45.
Search using a PCB with PROCSEQ

Fast Path Indexer/EP lets you insert, replace, or delete a segment through an alternate processing sequence method, with one exception. If you try to perform a non-retrieval process, the path calls must be in the original processing sequence. Fast Path Indexer/EP inverts the hierarchy only during retrieval processing.

**NOTE**

Fast Path Indexer/EP does not support path call insert, replace, or delete through an alternate processing sequence.

Assume that your application program issues a DL/I call that refers to a PCB containing the PROCSEQ keyword (meaning alternate sequence processing is to be performed). (For information about using the PROCSEQ keyword in your application PSB, see page 76.) Code the PCB as follows:

```
PCB DBDNAME=DIVSSD,PROCOPT=A,PROCSEQ=DIVSI4
```

The application program tries to retrieve all Student and Course segments in Area Code 888. It issues the following DL/I call:

```
CALL 'CBLTDLI' USING FUNC-GU
   DIVSSD-PCB
   USER-IO-AREA
   SSA-DIVSSD-QUAL
```

The qualified SSA in the program would look like the following example:

```
STUDENT *D-(SRCVALU  =888)
LOCN  *-
COURSE *-
```

When your application program successfully retrieves the Student segment, the following information is returned to the program:

I/O area: STUDENT AND COURSE SEGMENTS
Key feedback area: 888LOC#COURS1

The application program then issues the following GN call:

```
CALL 'CBLTDLI' USING FUNC-GN
   DIVSSD-PCB
   USER-IO-AREA
   SSA-DIVSSD-QUAL
```
The qualified SSA in the program would look like the following example:

```
STUDENT *D-(SRCVALU =888)
LOCN *-
COURSE *-
```

When your application program successfully retrieves the next Student and Course segment, the following information is returned to the program:

I/O area: STUDENT AND COURSE SEGMENTS  
Key feedback area: 888LOC#COURS2

### Search using a PCB with a stand-alone index

Assume that your application program issues a DL/I call against an index to retrieve duplicate data. The primary DEDB is not accessed. Code the PCB as follows:

```
PCB DBDNAME=DIVSI4,KEYLEN=99,PROCOPT=G
```

For this type of call, you must specify PROCOPT as a retrieval.

The application program tries to retrieve all Student segments with Student Area Code 111. The program issues the following DL/I call:

```
CALL 'CBLTDLI' USING FUNC-GU
DIVSI4-PCB
USER-IO-AREA
SSA-DIVSI4-QUAL
```

The qualified SSA in the program would look like the following example:

```
ROOTSEG *-(ROOTKEY >=1110000000000000*)
ROOTKEY <=111FFFFFFFFFFFFF)
```

A lowercase ‘o’ represents a low value; a lowercase ‘f’ represents a high value.

When your application program successfully retrieves the Student segment, the following information is returned to the program:

I/O area: POINTER SEGMENT OF THE INDEX  
Key feedback area: 111STUD#LOC#COURS1  
<-SUBSEQ field->
Processing an index as a stand-alone database

For retrieval only, Fast Path Indexer/EP lets you process your index as a stand-alone database. Fast Path Indexer/EP treats this type of retrieval like any other. If you process a partitioned SHISAM index or a HISAM index as a stand-alone database, ensure that your application programs know that the index is partitioned. The application programs process each partition as a separate database, so each partition would have to be a separate PCB in the PSB.

If you process your index as a stand-alone database, you can include duplicate data (DDATA) on the index. DDATA indicates that data residing on the primary DEDB is duplicated on the index.

You might want to include DDATA on the index so that you can process only the index to obtain specific data. Otherwise, you must process the entire primary DEDB, which would be time-consuming and costly.
Processing an index as a stand-alone database
Adding index functionality to the Fast Path environment

This chapter provides an overview of the steps that you must perform to define a primary DEDB and build PFX or IBM native indexes for the first time. Many of these steps are applied when you maintain Fast Path Indexer/EP indexes.

Conceptually, the Fast Path Indexer/EP creation process comprises the following tasks:

1. defining the primary DEDB, defining the index, and registering the index

   These actions are referred to as the index registration process.

2. building the index

3. implementing the index in IMS by refreshing the control blocks in the control region

Table 6 lists the required steps for using a secondary index database. The ‘Performed by’ column indicates whether the step is performed by using the Fast Path Indexer/EP product (PFX) or native IMS (IMS). Steps that are performed within IMS should be somewhat routine for users who are familiar with building and maintaining indexes to IMS full-function databases. The rightmost column lists where each step of the index registration process is discussed in more detail.

Table 6  Fast Path Indexer/EP usage process (part 1 of 2)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Performed by</th>
<th>Reference</th>
</tr>
</thead>
</table>
| 1    | Modify the primary DBD to include Fast Path Indexer/EP macros. | PFX          | • Chapter 4, “Fast Path Indexer/EP PFX index macros”  
|      |                                                  |              | • Chapter 6, “Primary DEDB and PFX index control block maintenance” |
### Table 6  Fast Path Indexer/EP usage process (part 2 of 2)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Performed by</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Run IMS DBDGEN for the primary DEDB.</td>
<td>IMS</td>
<td>Chapter 6, “Primary DEDB and PFX index control block maintenance”</td>
</tr>
<tr>
<td>3</td>
<td>Generate the index DBD.</td>
<td>IMS</td>
<td>Chapter 6, “Primary DEDB and PFX index control block maintenance”</td>
</tr>
<tr>
<td>4</td>
<td>Run IMS DBDGEN for indexes.</td>
<td>IMS</td>
<td>Chapter 6, “Primary DEDB and PFX index control block maintenance”</td>
</tr>
<tr>
<td>5</td>
<td>Create or update the PSB source.</td>
<td>IMS</td>
<td>Chapter 6, “Primary DEDB and PFX index control block maintenance”</td>
</tr>
<tr>
<td>6</td>
<td>Run IMS PSBGEN.</td>
<td>IMS</td>
<td>Chapter 6, “Primary DEDB and PFX index control block maintenance”</td>
</tr>
<tr>
<td>7</td>
<td>Run the PFXAGEN utility.</td>
<td>PFX</td>
<td>- Chapter 5, “Fast Path Indexer/EP PFXAGEN utility”</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Chapter 6, “Primary DEDB and PFX index control block maintenance”</td>
</tr>
<tr>
<td><strong>Build the index</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Execute the BUILD command to build an index.</td>
<td>PFX</td>
<td>Chapter 9, “Index build utility”</td>
</tr>
<tr>
<td><strong>Implement the index in IMS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Refresh the control blocks in the IMS control region.</td>
<td>IMS</td>
<td>Chapter 5, “Fast Path Indexer/EP PFXAGEN utility”</td>
</tr>
</tbody>
</table>
Implementing PFX index databases

This part presents the following topics:

Chapter 4
   Fast Path Indexer/EP PFX index macros. ........................................ 59

Chapter 5
   Fast Path Indexer/EP PFXAGEN utility ........................................ 73

Chapter 6
   Primary DEDB and PFX index control block maintenance. ............... 97
Fast Path Indexer/EP PFX index macros

This chapter discusses adding Fast Path Indexer/EP PFX index macros to the primary DBD. These macros define the SHISAM or HISAM indexes that you can build to be associated with a primary DEDB.

This chapter discusses the following topics:

Overview ................................................................. 59
PFXCHILD macro ....................................................... 60
  PFXCHILD keywords ............................................. 60
  PFXCHILD syntax diagrams ................................... 63
  PFXCHILD macro examples .................................. 63
PFXPART macro ....................................................... 63
  PFXPART keyword ............................................... 64
  PFXPART syntax diagram ....................................... 64
  PFXCHILD/PFXPART macro examples ................... 65
PFXXDFLD macro ..................................................... 67
  PFXXDFLD keywords ........................................... 67
  PFXXDFLD syntax diagrams ................................ 69
  PFXXDFLD macro example .................................. 69
PFXGEN macro ....................................................... 70
Creating multiple PFX index records from one source segment ................. 70

Overview

You must include the Fast Path Indexer/EP PFX index macros when you define the primary DEDB to the product by modifying its DBD (the primary DBD). The product includes the macros listed in Table 7.
The PFXCHILD macro statement is similar to the IMS LCHILD macro statement. PFXCHILD defines the segment name or names that Fast Path Indexer/EP uses in an SSA to retrieve data through the PFX index. Unlike the LCHILD macro, however, PFXCHILD includes the additional ACCESS keyword. The value that is specified for this keyword defines the structure (access method) for the PFX index, and can be combined with additional optional values. You must include the PFXCHILD macro statement when you define a primary DEDB. Place the PFXCHILD macro following the last IMS FIELD statement for the target segment.

PFXCHILD keywords

Table 8 describes the PFXCHILD keywords and parameters that are used for defining SHISAM or HISAM indexes. The keywords are listed in Table 8 in the order that they should appear on your macro statements.
PFXCHILD keywords

Table 8  PFXCHILD keywords and parameters (part 1 of 2)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| NAME    | (seg1,(indexdb1)) | Required.  
  
  *seg1* defines the segment names used in a SHISAM or HISAM index DBD. The name must be from one to eight characters in length, starting with an alphabetic character. Fast Path Indexer/EP will use this defined segment name in an SSA to retrieve the target segment through the SHISAM or HISAM index. The SHISAM or HISAM index DBD contains only one segment, which includes pointer and root information.  
  
  *indexdb1* specifies the DBD name of the SHISAM or HISAM index.  

| (seg1,(indexdb1,...,indexdbnnn)) | indexdb1,...,indexdbnnn | specifies the partitioned PFX index DBD names. A separate PFX index DBD must be created for each partitioned PFX index DBD name that is specified. Each partitioned PFX index database will have the same segment name as specified for *seg1*. The first partitioned PFX index DBD (*indexdb1*) will reflect the DBD name that is used in a PSB PROCSEQ statement.  
  
  This parameter has a length limit of 255 characters (HLASM restriction), including the segment name, PFX index DBD names, and punctuation. If the list of partitioned PFX index DBD names exceeds this limit, use the PFXPART macro to specify the additional partitioned PFX index DBD names (one DBD name per PFXPART statement). For more information about coding the PFXPART macro, see “PFXPART macro” on page 63. |
| ACCESS | SHISAM | Required to indicate that the PFX index defined by the *indexdbn* parameter uses the SHISAM access method. You must specify either SHISAM or HISAM on the ACCESS keyword. |
| (SHISAM, COMPACT) | | Required for an PFX index definition when the source segment is non-keyed.  
  
  The COMPACT parameter builds the PFX index database by using an alternate format (compact). The compact format (Figure 40 on page 113) is smaller than the standard index format (Figure 39 on page 112). If an application program accesses the PFX index as a stand-alone database, you must specify the appropriate format. |
| HISAM | | Required to indicate that the PFX index defined by the *indexdbn* parameter uses the HISAM access method. You must specify either HISAM or SHISAM on the ACCESS keyword. |
### Table 8  PFXCHILD keywords and parameters (part 2 of 2)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS (continued)</td>
<td>(HISAM, COMPACT)</td>
<td>Required for a PFX index definition when the source segment is non-keyed. The COMPACT parameter builds the PFX index database by using an alternate format (compact). The compact format (Figure 40 on page 113) is smaller than the standard index format (Figure 39 on page 112). If an application program accesses the PFX index as a stand-alone database, you must specify the appropriate format.</td>
</tr>
<tr>
<td>VERTICAL</td>
<td></td>
<td>Default. This parameter indicates a partitioned SHISAM or HISAM index that uses the vertical processing method.</td>
</tr>
<tr>
<td>LATERAL</td>
<td></td>
<td>Optional. This parameter indicates a partitioned SHISAM or HISAM index that uses the lateral processing method. This method processes data from the beginning of the partition until the end; at the end of the partition, Fast Path Indexer/EP issues a GB status code. If you do not specify ACCESS=LATERAL, the product defaults to using the vertical processing method.</td>
</tr>
<tr>
<td>ERRORACTION</td>
<td>ABEND</td>
<td>Default. Indicates that the application program terminates abnormally with an abend. Use the ABEND, ROLB, ROLB-xx, STATUSCODE, or STATUSCODE-xx parameter on the ERRORACTION keyword to specify the action to perform when errors occur during PFX index maintenance.</td>
</tr>
<tr>
<td></td>
<td>ROLB</td>
<td>Indicates that a ROLB call is issued to roll back changes made to the database since the last synchronization. The original status code indicating a PFX index maintenance error is returned to the application program.</td>
</tr>
<tr>
<td></td>
<td>ROLB-xx</td>
<td>Indicates that a ROLB call is issued to roll back changes made to the database since the last synchronization. The original status code indicating a PFX index maintenance error is returned to the application program. The status code (xx) is user-defined.</td>
</tr>
<tr>
<td></td>
<td>STATUSCODE</td>
<td>Indicates that the application call to the primary database has completed without error, but PFX index maintenance is terminated. This action might cause a PFX index to be inconsistent. The original status code indicating a PFX index maintenance error is returned to the application program.</td>
</tr>
<tr>
<td></td>
<td>STATUSCODE-xx</td>
<td>Indicates that the application call to the primary database has completed without error, but PFX index maintenance is terminated. This action might cause a PFX index to be inconsistent. The original status code indicating a PFX index maintenance error is returned to the application program. The status code (xx) is user-defined.</td>
</tr>
</tbody>
</table>
**PFXCHILD syntax diagrams**

Figure 15 illustrates the syntax of a PFXCHILD statement for a SHISAM to HISAM index.

![Syntax Diagram](image)

**PFXCHILD macro examples**

Figure 16 on page 63 shows how to use the PFXCHILD macro to define segment names in a SHISAM index. The list of parameters specified on the NAME keyword does not exceed the 255-character limit.

![Macro Example](image)

**PFXPART macro**

Because of a limitation in the current version of the High Level Assembler (HLASM), the NAME parameter on the PFXCHILD macro allows a maximum of 255 characters to be specified to list the target segment name and the partitioned PFX index DBD names. This restriction might be relieved in a future release of HLASM.

The PFXPART macro statement provides an alternate method to specify some or all of the partitioned PFX index DBD names for a partitioned SHISAM or HISAM index. Because of the 255-character HLSAM restriction for the PFXCHILD NAME parameter, the PFXPART macro must be used when you are defining a large number of PFX indexes. Optionally, the PFXPART macro can be used to list all of the partitioned PFX index DBD names.
You must specify only one keyword (NAME) with one PFX index name for each PFXPART macro that is defined in the primary DBD. All PFXPART macros are subordinate to the PFFCHILD macro and must be placed after the PFXCHILD definition in the primary DBD.

Examples of the two alternative methods of specifying partitioned PFX index DBD names are shown beginning in “PFXCHILD/PFXPART macro examples” on page 65.

**NOTE**

If you use the PFXPART macro to list all of the partitioned PFX index DBD names, do not specify the `indexdbd1,...,indexdbdnnn` values following the `seg1` parameter on the PFXCHILD NAME keyword.

---

**PFXPART keyword**

Table 9 describes the NAME keyword that can be specified on the PXFPART macro to identify partitioned PFX index DBD names for a PFX index that was defined as SHISAM or HISAM on the PFXCHILD macro.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td><code>indexdb</code></td>
<td><code>indexdb</code> specifies the partitioned PFX index DBD names. You must specify the <code>NAME=indexdb</code> syntax for each additional partitioned PFX index that you want to define. Each partitioned PFX index database will have the same segment name as defined in the PFXCHILD <code>NAME(seg1)</code> field. You can specify a total of 500 PFX index partitions for the primary DEDB.</td>
</tr>
</tbody>
</table>

---

**PFXPART syntax diagram**

Figure 17 illustrates the syntax of a PFXPART macro statement for defining partitioned SHISAM or HISAM indexes. The recursive arrow in the diagram indicates that the entire macro statement must be specified for each partitioned PFX index DBD name to be associated with the source target segment.

**Figure 17** PFXPART statement for defining partitioned PFX indexes

```
PFXPART  --  NAME=indexdb
```
PFXCHILD/PFXPART macro examples

Figure 18 shows how to code multiple PFXPART macro statements with the PFXCHILD macro to define additional partitioned PFX index DBD names in a SHISAM index. Some of the PFX index DBD names are defined on the NAME keyword on the PFXCHILD macro statement; the remaining PFX index DBD names are specified on PFXPART macro statements. A total of 40 PFX index partitions are specified for the target segment. This example circumvents the restriction of exceeding the 255-character limit on the PFXCHILD NAME parameter.

Figure 18  PFXCHILD/PFXPART macro example using the PFXPART macro to specify additional partitioned PFX index databases

<table>
<thead>
<tr>
<th>PFXCHILD NAME=(PFXROOT, (DIVSI4, DIVSI42, DIVSI43, X DIVSI44, DIVSI45, DIVSI46, X DIVSI47, DIVSI48, DIVSI49, X DIVSI410, DIVSI411, DIVSI412)), X ACCESS=(SHISAM, LATERAL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFXPART NAME=DIVSI413</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI414</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI415</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI416</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI417</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI418</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI419</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI420</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI421</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI422</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI423</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI424</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI425</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI426</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI427</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI428</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI429</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI430</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI431</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI432</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI433</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI434</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI435</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI436</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI437</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI438</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI439</td>
</tr>
<tr>
<td>PFXPART NAME=DIVSI440</td>
</tr>
</tbody>
</table>
Figure 19 shows how to use multiple PFXPART macro statements to define all of the partitioned PFX index DBD names to be associated with a target segment in a SHISAM index. A total of 35 PFX index partitions are specified. Only the target segment is specified on the PFXCHILD NAME keyword; the indexdbd1,...,indexdbdnnn syntax is omitted. Coding the macros in this manner eliminates the possibility of exceeding the character length limitation on the PFXCHILD NAME parameter.

**Figure 19  PFXCHILD/PFXPART macro example using the PFXPART macro to specify all partitioned PFX index databases**

```plaintext
PFXCHILD NAME=(PFXROOT), ACCESS=(SHISAM,LATERAL)
PFXPART NAME=DIVSI4
PFXPART NAME=DIVSI42
PFXPART NAME=DIVSI43
PFXPART NAME=DIVSI44
PFXPART NAME=DIVSI45
PFXPART NAME=DIVSI46
PFXPART NAME=DIVSI47
PFXPART NAME=DIVSI48
PFXPART NAME=DIVSI49
PFXPART NAME=DIVSI410
PFXPART NAME=DIVSI411
PFXPART NAME=DIVSI412
PFXPART NAME=DIVSI413
PFXPART NAME=DIVSI414
PFXPART NAME=DIVSI415
PFXPART NAME=DIVSI416
PFXPART NAME=DIVSI417
PFXPART NAME=DIVSI418
PFXPART NAME=DIVSI419
PFXPART NAME=DIVSI420
PFXPART NAME=DIVSI421
PFXPART NAME=DIVSI422
PFXPART NAME=DIVSI423
PFXPART NAME=DIVSI424
PFXPART NAME=DIVSI425
PFXPART NAME=DIVSI426
PFXPART NAME=DIVSI427
PFXPART NAME=DIVSI428
PFXPART NAME=DIVSI429
PFXPART NAME=DIVSI430
PFXPART NAME=DIVSI431
PFXPART NAME=DIVSI432
PFXPART NAME=DIVSI433
PFXPART NAME=DIVSI434
PFXPART NAME=DIVSI435
```
PFXXDFLD macro

The PFXXDFLD macro statement is similar to the IMS XDFLD macro statement. PFXXDFLD defines options for the source and target segments. When you define a primary DEDB, you must include the PFXXDFLD macro statement. Place the PFXXDFLD macro statement after the PFXCHILD macro statement in the primary DBD.

PFXXDFLD keywords

Table 10 describes the PFXXDFLD keywords and parameters. The keywords are listed in the order that they should appear on your macro statements.

Table 10  PFXXDFLD keywords and parameters (part 1 of 3)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>key1</td>
<td>Required. Defines the sequence field that Fast Path Indexer/EP will use in an SSA to retrieve the target segment through a SHISAM or HISAM index. SHISAM and HISAM indexes contain only one sequence field, which includes pointer and root information.</td>
</tr>
<tr>
<td>IKFNAME</td>
<td>fieldname</td>
<td>Optional. Defines the sequence (SEQ) field name to be used in the primary DBD for the PFX index database. This value is specified in an SSA for a DL/I call statement in the application program to access the PFX index as a stand-alone database. If this keyword is not specified, the value specified for the NAME keyword is used by default.</td>
</tr>
<tr>
<td>SRCH</td>
<td>field1, (field2,..., field5)</td>
<td>Required. Defines from one to five source segment fields that Fast Path Indexer/EP uses as the PFX index key. You can use any field defined by a FIELD macro statement that refers to the source segment. If you specify more than one field, you must enclose the field names within parentheses.</td>
</tr>
<tr>
<td>/CK</td>
<td></td>
<td>Optional. You can use the /CK, start, length command to indicate that the SRCH data should be obtained from the concatenated key from the root to the source segment. You must follow /CK with information about a start location, then provide a length. From the starting position, the specified length cannot exceed the concatenated key length. You can use as many as five concatenated fields; however, they must be enclosed in parentheses. Use the following format: SRCH=(field1,(/CK,1,15))</td>
</tr>
</tbody>
</table>
### PFXXDFLD keywords

#### Table 10  PFXXDFLD keywords and parameters (part 2 of 3)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| SUBSEQ  | field1,(field2,..., field5) | Optional. Defines from one to five segment fields that Fast Path Indexer/EP uses as subsequence data fields in the SHISAM or HISAM index. This value makes the database key unique.  

**Note:** Even though the field value for the SUBSEQ keyword will be merged with the field value of the SRCH keyword to create the PFX index segment key, the value for the SUBSEQ keyword cannot be specified as part of the search criteria in the SSA. The field value for the SUBSEQ keyword is not recognized by the application program unless it is processing the PFX index as a stand-alone database. |
| /CK     |           | You can use the `/CK, start, length` command to indicate that the subsequence data should be obtained from the concatenated key from the root to the source segment.  
You must follow `/CK` with information about a start location, then provide a length. From the starting position, the specified length cannot exceed the concatenated key length. You can use as many as five concatenated fields; however, they must be enclosed in parentheses. Use the following format:  

```
SUBSEQ=((/CK,1,15),field2....field5)
``` |
| NULLVAL | X'xx'     | Optional. Defines a hexadecimal value that Fast Path Indexer/EP should use for sparse indexing. This keyword does not include a specific parameter; you can use any self-defining term. For example, you could use NULLVAL=X'40'.  
Fast Path Indexer/EP does not perform indexing when each field of the PFX index source segment specified by the SRCH keyword has the value indicated by NULLVAL in every byte. |
| BLANK   |           | Optional. Defines a blank as the character that Fast Path Indexer/EP should use for sparse indexing. |
| ZERO    |           | Optional. Defines a binary zero as the character that Fast Path Indexer/EP should use for sparse indexing. |
| DDATA   | field1,(field2,..., field5) | Optional. Defines from one to five source segment fields that Fast Path Indexer/EP can use as duplicate data fields. You must have already defined these fields in the previous FIELD macro statement or statements that define the source segment.  
DDATA fields reside in the pointer segment, and the application program can refer to them while processing the PFX index as a stand-alone database. If you specify more than one field, you must enclose them in parentheses. |
| EXTRTN  | exitname  | Optional. Defines a user-written sparse exit routine that Fast Path Indexer/EP can call prior to inserting or replacing any PFX index segment. If the exit issues a nonzero return code, Fast Path Indexer/EP will not index the source segment. |
PFXXDFLD syntax diagrams

Figure 20 illustrates the syntax of a PFXXDFLD statement for a SHISAM or HISAM index.

**Figure 20** PFXXDFLD statement for SHISAM or HISAM index

```
PFXXDFLD NAME=key1, SRCH=(field1,(field2,...,field5)),
        SUBSEQ=(field1,(field2,...,field5)),
        DDATA=(field1,(field2,...,field5)),
        EXTRTN=exitname, NULLVAL=X’xx’,
        PARTEXIT=partitionexitname, SEGMENT=sourceseg
```

Table 10  PFXXDFLD keywords and parameters (part 3 of 3)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEGMENT</td>
<td>sourceseg</td>
<td>Optional. Defines the source segment if it is different from the target segment. If you use this parameter, you must use the current target segment or a segment that resides below the target.</td>
</tr>
<tr>
<td>PARTEXIT</td>
<td>partitionexitname</td>
<td>Required if the database uses partitioned indexes. Indicates that you want to use a user-written partitioned exit routine.</td>
</tr>
</tbody>
</table>

PFXXDFLD macro example

Figure 21 shows how to use the PFXXDFLD macro to define segments for a SHISAM or HISAM index.

**Figure 21** PFXXDFLD macro example for a SHISAM or HISAM index (part 1 of 2)

```
PFXXDFLD NAME=SRCVALU,SEGMENT=STUDENT,
          SRCH=(STUDAREA,(/CK,1,5)),
          DDATA=(STUDSNME,STUDFNME,STUDPHON)
```
PFXGEN macro

When you define a primary DEDB, you must include the PFXGEN macro statement. Place PFXGEN after the DBDGEN macro statement and before the FINISH statement in the primary DBD.

The PFXGEN macro statement collects information from the PFXCHILD and PFXXDFLD macro statements, then builds the PFX definition block. PFXGEN also places a CSECT named PFXREGI in the IMS DBDGEN output at the end of the primary DBD.

Figure 22 shows how to use the PFXGEN macro at the end of the primary DBD to build the PFX definition block.

Creating multiple PFX index records from one source segment

You can use multiple pairs of PFXCHILD and PFXXDFLD macros to create multiple PFX index records from the same source segment. Figure 23 illustrates the primary DBD changes that are required to create two or more PFX index records from one source segment. The multiple PFX index records must have the same accumulated length.

Figure 23 Creating multiple PFX index records from the same source segment (part 1 of 2)
To suppress any source pointer segments, you can use a sparse exit routine. For more information, see “Adding or modifying a sparse exit routine” on page 109.
Fast Path Indexer/EP PFXAGEN utility

This chapter provides detailed information about the Fast Path Indexer/EP PFXAGEN utility that you use instead of the IMS ACBGEN utility.

This chapter discusses the following topics:

Overview ................................................................. 74
Front-end process to ACBGEN ..................................... 74
PROCSEQ keyword ..................................................... 76
Shadow PCBs ............................................................. 76
PFXLIB allocation and usage ...................................... 77
Generating ACBs ....................................................... 78
DBD processing ......................................................... 79
PSB processing ........................................................ 80
Executing PFXAGEN .................................................. 82
JCL requirements ....................................................... 82
Control statement syntax ......................................... 85
Return codes .......................................................... 86
Report ................................................................. 87
PFXAGEN options ..................................................... 88
EXEC statement options ......................................... 88
DD statement options .............................................. 88
Dynamic allocation options .................................. 89
Changing PFXLIB online ........................................... 93
JCL requirements ..................................................... 94
DFSUOCU0 DD statements ...................................... 94
Overview

This chapter describes the batch utility, PFXAGEN, which provides a front-end process to IMS ACBGEN. PFXAGEN generates PFX-related application control blocks (ACBs) for DBD and program specification block (PSB) registration. PFX-related ACBs are used for subsequent processing of Fast Path Indexer/EP, both in the IMS online environment and in offline processing. You must run PFXAGEN as an integral step in the PFX index registration process; running IMS ACBGEN is not sufficient.

PFXAGEN generates the ACBs for PFX-related PSBs and database descriptions (DBDs), and for non-PFX PSBs and DBDs. PFXAGEN invokes IMS PSBGEN to update all PFX-related PSBs. Use of the PROCSEQ keyword and shadow PCBs are also discussed in this section.

Front-end process to ACBGEN

PFXAGEN invokes IMS ACBGEN without modifying the ACBGEN utility program. IMS ACBGEN is invoked in the same way it is invoked during a normal IMS ACBGEN procedure. You must use PFXAGEN so that it can perform a front-end process to IMS ACBGEN for PFX-related control blocks. PFXAGEN builds PFX control blocks and saves them in the PFXLIB data set. These control blocks are used only by Fast Path Indexer/EP, and are required to process PFX indexes.

By allowing PFXAGEN to obtain control before IMS ACBGEN, Fast Path Indexer/EP ensures that all applicable control blocks are updated and generated appropriately.

PFXAGEN identifies all PSBs that refer to a PFX-defined database. If you do not replace your existing IMS ACBGEN procedure with PFXAGEN, ensure that you do not accidentally run IMS ACBGEN for PFX-related PSBs and DBDs. If you run the IMS ACBGEN, it deactivates PFX for PFX-related PSBs and DBDs.

NOTE

BMC recommends that you replace your current IMS ACBGEN procedure with the PFXAGEN procedure to avoid out-of-sync situations that could occur between the control blocks that reside in PFXLIB and the ACBs that reside in ACBLIB.

PFXAGEN uses the DBDs and PSBs that were created during the index registration process to build the PFX control blocks that are stored in the PFXLIBx data sets. These control blocks contain information that Fast Path Indexer/EP needs to process the defined databases. PFXAGEN creates PSB entries for each PSB that refers to a defined primary DEDB or a PFX index. Figure 24 illustrates the PFX index registration process.
Figure 24  PFX index registration process flow

1. **source DBD**
2. **DBDGEN**
3. **DBDLIB**
4. **PFX index DBD**
   - **no**: **non-PFX DBD**
   - **yes**: modify PSB source
5. **PSBGEN**
6. **PSBLIB**
7. **PFXAGEN**

**Paths:**
- **ACBLIB**
- **PFXLIB**
- **PSBLIB**

**Notes:**
- ACBLIB, PSBLIB, PFXLIB are libraries that may or may not be used depending on the specific requirements.
- The flowchart illustrates the process of generating PFX indexes and their integration with other components.
**PROCSEQ keyword**

PFXAGEN lets you use the PROCSEQ keyword on the PCB that is associated with a primary DEDB. PFXAGEN processes and updates the Fast Path Indexer/EP control blocks and extracts the PROCSEQ keyword from the PCB.

To specify the PROCSEQ keyword in your application PSB, perform one of the following tasks:

- Specify the PFX index database name for the alternate processing sequence, as shown in the following example:

  ```
  PCB DBDNAME=DIVSSD,PROCOPT=A,PROCSEQ=DIVS14
  ```

- Specify a wildcard (*) for dynamic index selection, as shown in the following example:

  ```
  PCB DBDNAME=DIVSSD,PROCOPT=A,PROCSEQ=*  
  ```

  The wildcard allows PFXAGEN to process multiple PFX indexes when only one PCB within the PSB is specified.

  When you specify a wildcard, Fast Path Indexer/EP analyzes the first SSA for a retrieval call. The product determines which PFX index to use as follows:

  - If the segment named in the SSA is the target segment for a PFX index, and if the first field name specified in the SSA matches the name of the PFXXDFLD for the PFX index, Fast Path Indexer/EP uses that PFX index.

  - Otherwise, Fast Path Indexer/EP does not use a PFX index for the call.

**Shadow PCBs**

PFXAGEN modifies PSBs and ACBs with *shadow* PCBs for each PFX index that is associated with a primary DEDB. Shadow PCBs are PCBs that PFXAGEN uses for its PFX index processing. Shadow PCBs are visible only to Fast Path Indexer/EP and are transparent to the IMS application program. Fast Path Indexer/EP adds shadow PCBs for the following reasons:

- One shadow PCB is added for each PFX index that is associated with a primary DEDB. A PFX index could be referred to on multiple PCBs, with each PCB supporting a specific function. Fast Path Indexer/EP adds a shadow PCB for each defined PCB, even though all PCBs refer to the same primary DEDB and one PFX index.
One shadow PCB is added for each PFX-related PCB that contains the PROCSEQ keyword. Fast Path Indexer/EP does this so that it can maintain database positioning for all PCBs.

Shadow PCBs include the PCBNAME keyword and the LIST=NO parameter. LIST=NO prevents application programs from being affected by the additional shadow PCBs. This PCB includes PROCOPT=GO.

---

**NOTE**

PFXAGEN uses PCBNAME and LIST=NO to determine whether a shadow PCB is being reprocessed. If you do not modify these parameters, PFXAGEN processes shadow PCBs as normal PCBs. Do not modify PFX-generated PCBs through any method other than running PFXAGEN.

---

For information about how to define your primary DEDB, see Chapter 4, “Fast Path Indexer/EP PFX index macros.” For maintenance information, see Chapter 6, “Primary DEDB and PFX index control block maintenance.”

---

**PFXLIB allocation and usage**

Fast Path Indexer/EP maintains information about the primary DEDB and its associated PFX indexes in a partitioned data set named PFXLIB. The PFXLIB library contains members describing the primary DEDB, all associated PFX indexes, and any PSB that references them.

---

**PFXLIB allocation**

PFXLIB is allocated as a load library with appropriate attributes similar to the PSB or DBD libraries. Members are link-edited into PFXLIB by PFXAGEN during index registration. After registration, these members are referenced by MVS as load modules that allow Fast Path Indexer/EP to use the MVS load facility. PFXLIB usage is similar to ACBLIB usage and should not be allocated with extents.

---

**PFXLIB contents**

Although the member names are the same as those of the associated DBDs and PSBs, PFXLIB does not contain DBDs or PSBs. PFXLIB contains a representation of how the application views the database. Database registration members have a member for the primary DEDB and all of its defined PFX indexes. The primary DEDB member describes the DEDB, segments indexed, and fields used within the segment. The primary DEDB member also contains information about the uses of subsequence fields and duplicate data fields. The PFX index members contain information about
the PFX index and the hierarchical information about the primary DEDB. The PSB registration members (PSBMs) contain information about the PSB in general and each database PCB in the PSB. The PSBMs tell Fast Path Indexer/EP which PCB to use for index maintenance and which PCB is an index PCB.

**PFXLIB usage**

PFXLIB is used in the IMS control region and the DLISAS region. In both regions, PFXLIB is allocated to DD statements in the same manner as ACBLIB, PFXLIBA, and PFXLIBB. Similar allocation ensures compatibility with online changes. The active PFXLIB will always be the same as the active ACBLIB. When IMS reads from ACBLIBA, Fast Path Indexer/EP loads any associated members from PFXLIBA. Because these members can be loaded from the control region or the DLISAS region, just as ACBLIB members can be loaded, PFXLIBA and PFXLIBB must point to the same data set in the IMS control region and the DLISAS region. Fast Path Indexer/EP examines both regions at IMS initialization to ensure that these data sets are the same and to prevent their usage if they are not the same.

To help prevent the control region and DLISAS region from allocating different PFXLIBs, PFXLIB can be dynamically allocated by using the DFSMDA dynamic allocation macro. By dynamically producing a member for PFXLIBA and PFXLIBB, and by removing the DD statements from the IMS control region and the DLISAS region, Fast Path Indexer/EP dynamically allocates the active PFXLIB.

At IMS initialization, Fast Path Indexer/EP loads all database registration entries but does not load PSBMs. After IMS initialization, Fast Path Indexer/EP is invoked when an ACB is read by IMS and the appropriate PSBM is loaded. If the read is for a registered DBD, the registration blocks are refreshed. Before loading any blocks, Fast Path Indexer/EP ensures that the PFXLIB from which it will load matches the active ACBLIB. If the PFXLIB matches the active ACBLIB, Fast Path Indexer/EP does not switch libraries. This process allows Fast Path Indexer/EP to be sensitive to any online change without having to be invoked during online change. After an online change, a PFXLIB switch will not take place until the next ACBLIB access is performed by IMS.

**Generating ACBs**

PFXAGEN is a utility that runs as a front-end process to IMS ACBGEN. PFXAGEN processes the DBD or PSB that you specify on the BUILD control statement to determine whether the DBD or the PSB refers to any PFX-related control blocks.

If PFXAGEN determines that a specified DBD or PSB refers to any Fast Path Indexer/EP-related control blocks, it determines whether the DBD or PSB has been updated since the last PFXAGEN run. If an update has occurred, PFXAGEN updates PFXLIB with the new information.
IMS ACBGEN is run for the DBD or PSB that you specified on the BUILD control statement, even if PFXAGEN does not find updated PFX information.

PFXAGEN can perform the following types of processing:

- DBD processing
- PSB processing

Both methods can call each other during their processing. If you specify BUILD DBD, PFXAGEN performs DBD processing; if you specify BUILD PSB, PFXAGEN performs PSB processing. Both methods are described in this section.

## DBD processing

PFXAGEN performs the following steps during DBD processing:

1. **PFXAGEN checks DBD for PFX control blocks.**

   PFXAGEN checks the DBD that is specified on the BUILD control statement to determine whether it contains PFX control blocks. These control blocks would have been generated from the PFX macros specified during the IMS DBDGEN step in the PFX index registration process.

2. **PFXAGEN checks PFXLIB.**

   If PFXAGEN does not find PFX control blocks in ACBLIB, it checks PFXLIB to determine whether an entry exists for the specified DBD. If an entry exists, PFXAGEN deletes it and any PSB information that refers to this DBD. If no entry exists, PFXAGEN does not check this DBD any further.

   If PFXAGEN finds PFX control blocks in ACBLIB, it compares these control blocks to the entries in PFXLIB. If PFXAGEN determines that the control blocks differ from the entries in PFXLIB, PFXAGEN proceeds to step 3. If no differences are found, PFXAGEN proceeds to step 4.

3. **PFXAGEN updates PFXLIB.**

   PFXAGEN updates PFXLIB with the new information about the specified PFX DBD.
4. PFXAGEN scans ACBLIB for related PSBs.

PFXAGEN checks all PSBs contained in ACBLIB. PFXAGEN then narrows the search and focuses only on the PFX-related PSBs to determine whether any of these PSBs refer to the specified DBD. If PFXAGEN finds a PFX PSB that refers to the specified DBD, PFXAGEN calls the PSB processing routine to determine whether any PSB information has changed. For information about this processing method, see “PSB processing.”

PFXAGEN searches for more control statements or runs IMS ACBGEN as illustrated in Figure 25 on page 81.

**PSB processing**

PFXAGEN performs the following steps during PSB processing:

1. PFXAGEN checks the PSB for PFX-related PCBs.

   PFXAGEN checks the PSB specified on the BUILD control statement. If PFX determines that no PFX PCBs exist, it does not check this PSB any further. If PFXAGEN determines that any of these PCBs refer to PFX, PFXAGEN proceeds to step 2.

2. PFXAGEN compares the PSB to PFXLIB.

   PFXAGEN checks PFXLIB to determine whether the PSB has been updated since the last PFXAGEN run. If the PSB has not been updated, PFXAGEN does not check this PSB any further. If PFXAGEN determines that the PSB has been updated, PFXAGEN proceeds to step 3.

3. PFXAGEN checks related DBDs.

   PFXAGEN calls the DBD processing routine to determine whether IMS ACBGEN needs to be ran, and PFXAGEN determines whether any DBD information has changed. For more information about this processing method, see “DBD processing” on page 79.

4. PFXAGEN adds shadow PCBs.

   PFXAGEN ensures that the necessary shadow PCBs are in place for the specified PSB. PFXAGEN then places this updated PSB into the PSBLIB specified on the SYSLMOD DD statement.
5. PFXAGEN updates the PSBSRC data set.

PFXAGEN places a copy of the updated PSB source statements in the PSBSRC data set.

6. PFXAGEN searches for more control statements or runs IMS ACBGEN as illustrated in Figure 25.

When PFXAGEN completes its processing, the following output is produced:

- updated PFXLIB (if needed)
- updated PSB source member
- modified PSB, assembled and linked to the appropriate libraries
- updated ACB for the requested control blocks and all PFX-related control blocks
- activity list

Figure 25 illustrates the inputs and outputs of the PFXAGEN process.
Executing PFXAGEN

To execute PFXAGEN, you must use the PFXAGEN module that BMC supplied.

JCL requirements

Execute PFXAGEN as shown in Figure 26.

Figure 26  PFXAGEN sample JCL

```
EXEC PGM=PFXAGEN,REGION=4096K
```

The following statements are required or recommended for PFXAGEN:

**EXEC**

Required. Defines the PFXAGEN execution utility.

The minimum recommended region size is 4096 KB, as shown in the following example:

```
EXEC PGM=PFXAGEN,REGION=4096K
```
**STEPLIB DD**

Required. Defines the libraries containing the load modules that must be present to run PFXAGEN. This library contains the dynamic allocation members, if dynamic allocation is used. For more information, see “Dynamic allocation DD statements” on page 90.

**DFSRESLB DD**

Required. Defines the IMS RESLIB library.

**IMS DD**

Required. Defines the IMS PSB and DBD load libraries. If you specify PSB=ALL, the PSBLIB must appear first in the concatenation.

**PFXLIB DD**

Defines the staging PFXLIB. You must specify either PFXLIB or one of the following DD statements:

- MODSTAT
- OLCSTAT

**IMSACB DD**

Required. Defines the ACB library that contains the generated ACBs.

**SYSIN DD**

Required. Defines the data set that PFX uses to place the generated input to IMS ACBGEN. This data set must reside on DASD and must be a sequential file. If you supply data control block (DCB) information for this statement, specify RECFM=FB and LRECL=80.

**SYSLIB DD**

Required, but can be dynamically allocated. Defines the SYSLIB concatenation used by the assembler to find IMS and Fast Path Indexer/EP macros to regenerate a PSB with shadow PCBs. This library must contain the IMS MACLIBs.

**SYSLIN DD**

Required, but can be dynamically allocated. This two-cylinder temporary file is used by the assembler and the linkage editor to hold the unlinked code for any PSB modified during the PFXAGEN process. When SYSLIN is dynamically allocated, it is deleted automatically after use.
SYSUT1 DD

Required, but can be dynamically allocated. This two-cylinder temporary file is used by the assembler as a work file. When SYSUT1 is dynamically allocated, it is deleted automatically after use.

WORK001 DD

Required, but can be dynamically allocated. This two-cylinder temporary file is used by PFXAGEN to hold the modified PSB to be assembled and link-edited. When WORK001 is dynamically allocated, it is deleted automatically after use.

SYSPRINT DD

Required. Defines the SYSOUT file from the assembler, the linkage editor, and IMS ACBGEN.

PFXPRINT DD

Required only if you want to catalog your log output. The PFXPRINT DD statement identifies the processing log output data set. If the PFXPRINT DD statement is not specified in the JCL, Fast Path Indexer/EP dynamically allocates the data set using the following DCB characteristics:

<table>
<thead>
<tr>
<th>RECFM</th>
<th>LRECL</th>
<th>BLKSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBA</td>
<td>137</td>
<td>4096</td>
</tr>
</tbody>
</table>

The output data set must be a standard SYSOUT or sequential data set. The record format (RECFM) might specify fixed or variable length records, blocked or unblocked, and might include ANSI carriage control.

The logical record length (LRECL) can be any length. If an output record exceeds the LRECL, Fast Path/EP attempts to split the record into multiple lines along word boundaries. If an output record is shorter than the LRECL, the record is padded with trailing blanks.

The block size (BLKSIZE) can be any value appropriate for the LRECL and RECFM.

If you specify ANSI carriage control, a control character is generated for each logical record. If you do not specify ANSI carriage control, the Fast Path/EP Series product generates blank lines to simulate any carriage control function requested.
Control statement syntax

This section describes the PFXAGEN control statements and keywords. You can code Fast Path Indexer/EP control statements and keywords by using a free format in columns 1 through 71.

If you begin specifying data for a keyword on one line and need to continue to the next line, perform the following steps:

1. On the current line, include a closing parentheses mark before column 71.
2. On the next line, duplicate the same keyword for which you are still specifying data.
3. On the new line, specify the remaining data. An example is shown in Figure 27.

Figure 27  Continuing keyword parameters on an additional line

| BUILD | PSB=(x1,x2,x3,x4) |
|       | PSB=(x5,x6,x7,x8) |

Figure 28 shows a sample PFXAGEN control statement.

Figure 28  PFXAGEN control statements and keywords

//SYSIN DD *
BUILD  PSB=x
       =ALL
       =(x,x,x)
       DBD=x
       =(x,x,x)
DELETE PSB=x
       =ALL
       =(x,x,x)
       DBD=x
       =(x,x,x)

Table 11 describes the control statements and keywords that are valid for PFXAGEN.
### Table 11  Description of PFXAGEN control statements and keywords

<table>
<thead>
<tr>
<th>Control Statement</th>
<th>Keyword</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BUILD</strong></td>
<td></td>
<td></td>
<td>Required. Build blocks for the indicated control blocks (PSB or DBD).</td>
</tr>
<tr>
<td></td>
<td>PSB</td>
<td></td>
<td>Build blocks for one or more PSBs.</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>Build blocks for this PSB.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>Build blocks for all PSBs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x,x,x)</td>
<td>Build blocks for more than one PSB. There is no maximum number that you can specify.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DBD</td>
<td></td>
<td>Build blocks for one or more DBDs.</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>Build blocks for this DBD.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x,x,x)</td>
<td>Build blocks for more than one DBD. There is no maximum number that you can specify.</td>
<td></td>
</tr>
<tr>
<td><strong>DELETE</strong></td>
<td></td>
<td></td>
<td>Optional. Delete blocks for the indicated control blocks (PSB or DBD).</td>
</tr>
<tr>
<td></td>
<td>PSB</td>
<td></td>
<td>Delete blocks for one or more PSBs.</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>Delete blocks for this PSB.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>Delete blocks for all PSBs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x,x,x)</td>
<td>Delete blocks for more than one PSB. There is no maximum number that you can specify.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DBD</td>
<td></td>
<td>Delete blocks for one or more DBDs.</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>Delete blocks for this DBD.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x,x,x)</td>
<td>Delete blocks for more than one DBD. There is no maximum number that you can specify.</td>
<td></td>
</tr>
</tbody>
</table>

**Return codes**

PFXAGEN returns codes from the following sources:

- assembler
- linkage editor
- IMS ACBGEN
- PFXAGEN

The highest return code value acts as the final return code.
The Fast Path Indexer/EP ACBGEN Utility report (Figure 29) lists the output for the assembler, the linkage editor, the IMS ACBGEN procedure, and the PFXAGEN procedure. You control the output by specifying specific keywords and values in the PFXAGEN JCL.

Figure 29  Fast Path Indexer/EP ACBGEN utility report (part 1 of 2)

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILD    DBD=PFXTST1 0050 1023</td>
<td></td>
</tr>
</tbody>
</table>

BMC34367I DEFINITION BLOCK PFXTST1 ADDED
BMC34367I DEFINITION BLOCK PFXTIN01 ADDED
BMC34367I DEFINITION BLOCK PFXTIN02 ADDED
BMC34367I DEFINITION BLOCK PFXTIN03 ADDED
BMC34367I DEFINITION BLOCK PFXTIN04 ADDED
BMC34367I DEFINITION BLOCK PFXTIN05 ADDED
BMC34367I DEFINITION BLOCK PFXTIN06 ADDED
BMC34367I DEFINITION BLOCK DIVSSD ADDED
BMC34367I DEFINITION BLOCK DIVSI1 ADDED
BMC34367I DEFINITION BLOCK DIVSI2 ADDED
BMC34367I DEFINITION BLOCK DIVSI3 ADDED
BMC34367I DEFINITION BLOCK DIVSI4 ADDED
BMC34390I PSB REGISTRATION ENTRY DIVSBMP REPLACED
BMC34374I SOURCE UPDATE COMPLETE FOR PFXP0002
BMC34375I ASSEMBLER COMPLETE FOR PSB PFXP0002. RETURN CODE 0000
BMC34376I LINKAGE/EDITOR COMPLETE FOR PSB PFXP0002. RETURN CODE 0000
BMC34374I SOURCE UPDATE COMPLETE FOR PFXP0003
BMC34375I ASSEMBLER COMPLETE FOR PSB PFXP0003. RETURN CODE 0000
BMC34376I LINKAGE/EDITOR COMPLETE FOR PSB PFXP0003. RETURN CODE 0000
BMC34374I SOURCE UPDATE COMPLETE FOR PFXP0004
BMC34375I ASSEMBLER COMPLETE FOR PSB PFXP0004. RETURN CODE 0000
BMC34376I LINKAGE/EDITOR COMPLETE FOR PSB PFXP0004. RETURN CODE 0000
BMC34374I SOURCE UPDATE COMPLETE FOR PFXP0005
BMC34375I ASSEMBLER COMPLETE FOR PSB PFXP0005. RETURN CODE 0000
BMC34376I LINKAGE/EDITOR COMPLETE FOR PSB PFXP0005. RETURN CODE 0000
BMC34374I SOURCE UPDATE COMPLETE FOR PFXP0006
BMC34375I ASSEMBLER COMPLETE FOR PSB PFXP0006. RETURN CODE 0000
BMC34376I LINKAGE/EDITOR COMPLETE FOR PSB PFXP0006. RETURN CODE 0000
BMC34374I SOURCE UPDATE COMPLETE FOR PFXP0007
BMC34375I ASSEMBLER COMPLETE FOR PSB PFXP0007. RETURN CODE 0000
BMC34376I LINKAGE/EDITOR COMPLETE FOR PSB PFXP0007. RETURN CODE 0000
BMC34361I ACBGEN PROCESS COMPLETE. RETURN CODE 0004
BMC34390I PSB DEFINITION ENTRY PFXP0002 ADDED
BMC34390I PSB DEFINITION ENTRY PFXP0003 ADDED
BMC34390I PSB DEFINITION ENTRY PFXP0004 ADDED
BMC34390I PSB DEFINITION ENTRY PFXP0005 ADDED
PFXAGEN options

PFXAGEN allows execution through the following options:

- EXEC statement
- DD statements
- dynamic allocation

EXEC statement options

You can invoke PFXAGEN as an OS-callable program by specifying PGM=PFXAGEN.

DD statement options

The following optional DD statements can be used with PFXAGEN:

MODSTAT DD / OLCSTAT DD

You must specify either the PFXLIB DD statement or one of the following DD statements:

- MODSTAT DD, which defines the active IMS ACB library. This library will also be the active PFXLIBx library (PFXLIBA or PFXLIBB). If the MODSTAT DD statement is used, the PFXLIBA DD and PFXLIBB DD statements are also required but can be dynamically allocated.

- OLCSTAT DD, which functions the same as the MODSTAT DD statement but can be used for the IMS global online change function. The OLCSTAT DD statement is selected over the MODSTAT DD statement if both are present.

PFXLIBA DD / PFXLIBB DD

Defines the active PFXLIBx library (PFXLIBA or PFXLIBB) used by the IMS control region.
Dynamic allocation options

**NOTE**
MODSTAT DD, OLCSTAT DD, and PFXLIBx DD statements can be dynamically allocated. For more information, see “Dynamic allocation DD statements” on page 90.

**PSBSRC DD**
Defines output that is used only by PFXAGEN. The PSBSRC data set enables PFXAGEN to disassemble the PSB source for PFX-related PSBs. PFXAGEN returns the PSB source that contains the PROCSEQ keyword.

If you run an IMS PSBGEN later, PFX can import this PSB source, which contains PROCSEQ. This is the source that must be used; a disassembly of the PSBLIB would not be sufficient because it would not include the PROCSEQ keyword.

**WORK002 DD**
Defines a work PDS that PFXAGEN uses for control block processing. Use WORK002 when 250 or more PSBs are being built. WORK002 increases processing time but reduces virtual memory requirements. It must be large enough to contain PFXLIB.

Dynamic allocation options

Figure 30 shows PFXAGEN execution using dynamic allocation for SYSLIB, MODSTAT/OLCSTAT, PFXLIBA, and PFXLIBB.

**NOTE**
If you do not want to use dynamic allocation, see “Executing PFXAGEN” on page 82.

Figure 30  PFXAGEN sample JCL using dynamic allocation (part 1 of 2)

```plaintext
//***********************************************************************
//*           PFXAGEN with DYNAMIC ALLOCATION *                        
//***********************************************************************
//**
//**PFXAGEN JOB (xxxx),'PFXAGEN',MSGLEVEL=(2,0),
// NOTIFICATION=xxx,MSGCLASS=Y,CLASS=Q
//**BUILD EXEC PGM=PFXAGEN,REGION=4096K
//**STEPLIB DD DSN=SMTPS.PXS.LOAD,DISP=SHR
//** DD DSN=IMSVS.RESLIB,DISP=SHR<-- contains MDA members
//**DFSRESLIB DD DSN=IMSVS.RESLIB,DISP=SHR
//** IMS DD DSN=IMSVS.PSBLIB,DISP=SHR
//** DD DSN=IMSVS.DBDLIB,DISP=SHR
//**IMSACB DD DSN=IMSVS.ACBLIB,DISP=SHR
```
Dynamic allocation options

Figure 30  PFXAGEN sample JCL using dynamic allocation (part 2 of 2)

```
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSIN DD *
BUILD PSB=ALL
/*
```

Dynamic allocation DD statements

If you supply the DD statements in your JCL, PFXAGEN uses the specified data sets. Or, you can use the standard IMS dynamic allocation macro, DFSMDA, and create modules to dynamically allocate the data sets.

When using dynamic allocation, Fast Path Indexer/EP will search for the appropriate DFSMDA members in the following order:

1. in the data sets allocated to the STEPLIB DD statement
2. in the data sets allocated in LINKLIST

For the DD statements used by PFXAGEN, see “JCL requirements” on page 82.

PSB-generation DD statement

The following DD statement is used for PSB generation:

**SYSLIB DD**

This data set is used by the assembler during PSB generation and is called by PFXAGEN to regenerate a PSB with additional PCBs. SYSLIB describes the input macro library to the assembler process. This PDS must contain the macro statements for the PSB assembly process and the PFX macros needed by the assembler.

PFX registration DD statements

PFXAGEN tries to find a PFXLIB DD statement that is allocated. If the PFXLIB DD statement is present in the JCL, PFXAGEN uses it. If PFXLIB DD is not present in the JCL, PFXAGEN tries first to locate the OLCSTAT DD statement to determine the active PFXLIBx (PFXLIBA or PFXLIBB). If the OLCSTAT DD is not found, PFXAGEN tries to locate the MODSTAT DD statement to determine the active PFXLIBx (PFXLIBA or PFXLIBB). OLCSTAT, MODSTAT, PFXLIBA, and PFXLIBB can be dynamically allocated by using the DFSMDA macro.
Dynamic allocation options

The following DD statements are used for PFX registration:

**OLCSTAT**

The OLCSTAT DD is used by PFXAGEN when PFXLIB is not present to determine the active PFXLIBx (PFXLIBA or PFXLIBB) based on the active ACBLIB. OLCSTAT is used for the global online change function. If there is a DFSMDA member in the STEPLIB/LNKLIST for OLCSTAT, PFXAGEN opens and reads the associated data set, and then looks for the appropriate DD statement (PFXLIBA or PFXLIBB).

**MODSTAT**

The MODSTAT data set is used by PFXAGEN when PFXLIB is not present to determine the active PFXLIBx (PFXLIBA or PFXLIBB) based on the active ACBLIB. If there is a DFSMDA member in STEPLIB/LINKLIST for MODSTAT, PFXAGEN opens and reads the associated data set, then looks for the appropriate DD statement, PFXLIBA or PFXLIBB.

**PFXLIBA**

If the OLCSTAT/MODSTAT DD statement is present in the JCL and ACBLIBA is the active ACBLIB, PFXAGEN looks for the PFXLIBA DD statement. If PFXLIBA DD is not present in the JCL, PFXAGEN allocates the PFXLIBA data set to hold any PFX registration entries that are created or modified during this run of PFXAGEN.

**PFXLIBB**

If the OLCSTAT/MODSTAT DD statement is present in the JCL and ACBLIBB is the active ACBLIB, PFXAGEN looks for the PFXLIBB DD statement. If PFXLIBB DD is not present in the JCL, PFXAGEN allocates the PFXLIBB data set to hold any PFX registration entries created or modified during this run of PFXAGEN.

**Temporary DD statements**

When temporary files are dynamically allocated, they are deleted automatically after use. The following DD statements are temporary:

**SYSLIN DD**

This temporary file is used by the assembler and the linkage editor to hold the unlinked object code for any PSB modified during the PFXAGEN process.

**SYSUT1 DD**

This temporary file is used by the assembler as a work file.
Dynamic allocation options

WORK001 DD

This temporary file is used by PFXAGEN to hold the modified PSB to be assembled and link-edited.

**DFSMDA macro sample**

Use the DFSMDA macro to create members for each DD statement in STEPLIB. Figure 31 illustrates a sample DFSMDA macro.

### Figure 31  DFSMDA macro sample

```plaintext
DFSMDA TYPE=INITIAL
  DFSMDA TYPE=DATABASE,DBNAME=SYSLIB
  DFSMDA TYPE=DATASET,DSNAME=IMSVS.MACLIB,DDNAME=SYSLIB,DISP=SHR *
  DFSMDA TYPE=DATABASE,DBNAME=MODSTAT
  DFSMDA TYPE=DATASET,DSNAME=IMSVS.MODSTAT,DDNAME=MODSTAT,DISP=SHR *
  DFSMDA TYPE=DATABASE,DBNAME=OLCSTAT
  DFSMDA TYPE=DATASET,DSNAME=IMSVS.OLCSTAT,DDNAME=OLCSTAT,DISP=SHR *
  DFSMDA TYPE=DATABASE,DBNAME=PFXLIBA
  DFSMDA TYPE=DATASET,DSNAME=PFX.PFXLIBA,DDNAME=PFXLIBA,DISP=SHR *
  DFSMDA TYPE=DATABASE,DBNAME=PFXLIBB
  DFSMDA TYPE=DATASET,DSNAME=PFX.PFXLIBB,DDNAME=PFXLIBB,DISP=SHR *
DFSMDA TYPE=FINAL
END
```

**Dynamic allocation sequence**

PFXAGEN attempts to dynamically allocate members in the following order:

1. PFXAGEN checks the JCL for the PFXLIB DD statement, the OLCSTAT DD statement, or the MODSTAT DD statement.

2. If none of the DD statements listed above are present in the JCL, PFXAGEN checks STEPLIB/LINKLIST for the dynamic allocation members for OLCSTAT DD. If the OLCSTAT DD is not found, PFXAGEN checks for the dynamic allocation members for MODSTAT DD.

3. If the OLCSTAT DD statement is used, PFXAGEN checks the JCL for the PFXLIBA DD statement and the PFXLIBB DD statement. If PFXLIBA DD and PFXLIBB DD are not present in the JCL, PFXAGEN checks STEPLIB/LINKLIST for the dynamic allocation members.
4. If the MODSTAT DD statement is used, PFXAGEN checks the JCL for the PFXLIBA DD statement and the PFXLIBB DD statement. If PFXLIBA DD and PFXLIBB DD are not present in the JCL, PFXAGEN checks STEPLIB/LINKLIST for the dynamic allocation members.

5. PFXAGEN checks the JCL for the SYSLIB DD statement. If SYSLIB DD is not present in the JCL, PFXAGEN checks STEPLIB/LINKLIST for the dynamic allocation members.

6. PFXAGEN checks the JCL for the following DD statements: SYSLIN, SYSUT1, SYSLMOD, and WORK001. If SYSLIN DD, SYSUT1 DD, or WORK001 DD are not present in the JCL, PFXAGEN dynamically allocates temporary work data sets. It is recommended to not specify the SYSLMOD DD statement so that PFXAGEN can dynamically allocate the appropriate PFXLIB data set. PFXAGEN tries allocating to UNIT=VIO. If UNIT=VIO is unavailable, PFXAGEN allocates to UNIT=SYSALLDA.

Changing PFXLIB online

When you create a new ACB in the ACBLIB after executing PFXAGEN, you must also migrate the new member or members from the staging PFXLIB to the inactive online PFXLIB (PFXLIBA or PFXLIBB) before issuing the IMS online change commands. To migrate the new PFXLIB members from the staging PFXLIB to the online data set, you must execute the IMS Online Change Copy utility (DFSUOCU0).

Figure 32 shows the inputs and outputs of the online PFXLIB change process.

Figure 32  DFSUOCU0 inputs and outputs
When you have executed DFSUOCU0, you must issue the IMS online commands to execute the online change of the ACBLIB. Executing the IMS online commands will switch the inactive ACBLIB and will switch the inactive PFXLIB when the first transaction is scheduled after the change completes.

**JCL requirements**

Execute DFSUOCU0 as a standard OS job. To utilize this program, create a job by using the sample JCL shown in Figure 33. A model of this JCL is contained the member named PFXOLC in the Fast Path/EP sample library.

**Figure 33 DFSUOCU0 sample JCL**

```
COPY     EXEC PGM=DFSUOCU0,PARM=(ACB,S,U)  
STEPLIB  DD DSN=BMC.PFX.LOAD,DISP=SHR        <== must be authorized 
//      DD DSN=IMSVS.RESLIB,DISP=SHR
IMSACB   DD DSN=BMC.PFXLIB,DISP=SHR          <== staging library
IMSACBA  DD DSN=BMC.PFXLIBA,DISP=SHR         <== online library
IMSACBB  DD DSN=BMC.PFXLIBB,DISP=SHR         <== online library
SYSPRINT DD SYSOUT=*                          
MODSTAT  DD DSN=IMSVS.MODSTAT,DISP=SHR       
*/ or
OLCSTAT  DD DSN=IMSVS.OLCSTAT,DISP=SHR       
COPYCTL  DD UNIT=WORK,DISP=(NEW,DELETE),     
//      SPACE=(80,(100,100),RLSE),            
//      DCB=BLKSIZE=80,LRECL=80,RECFM=FB     
/ */
```

**DFSUOCU0 DD statements**

The following statements are valid with DFSUOCU0:

**EXEC**

Required. You must specify PGM=DFSUOCU0. The recommended region size is 4096 KB.

**STEPLIB DD**

Required. Defines the load library you need to run DFSUOCU0. This load library must be APF authorized.

**IMSACB DD**

Required. Defines the IMS PFXLIB data set that will be used as the input to IIEBCOPY.
IMSACBA DD

Required. Defines the PFXLIB to the IMS control region and DLISAS regions with the PFXLIBA DD statement.

IMSACBB DD

Required. Defines the PFXLIB to the IMS control region and DLISAS regions with the PFXLIBB DD statement.

SYSPRINT DD

Required. Defines the output of the reports generated by IEBCOPY that are routed to the output stream. If you supply DCB information for this statement, specify RECFM=FBA and LRECL=121.

MODSTAT DD

Required. Defines the IMS online change primary control data set.

MODSTAT2 DD

Optional. Defines the IMS online change secondary control data set.

OLCSTAT DD

Required in a global online change environment. Defines the IMS online change primary control data set. Using OLCSTAT also requires that the third parameter be specified as ‘G’ instead of ‘U’.

COPYCTL DD

Required. Defines a control statement input data set.
Primary DEDB and PFX index control block maintenance

This chapter describes how to use Fast Path Indexer/EP to maintain control blocks that define DEDBs and their associated PFX indexes.

This chapter discusses the following topics:

- Overview ................................................................. 98
- Online IMS maintenance ........................................... 98
- DBD maintenance ...................................................... 99
- Adding a PFX index ................................................... 99
  - Registering the PFX index ......................................... 99
  - Building the PFX index ............................................ 104
  - Implementing the PFX index in IMS ............................ 104
- Deleting PFX index registration .................................. 105
- Updating primary DBDs ............................................. 106
  - Updating DBDs that affect PFX indexes ...................... 106
  - Updating primary DBDs only .................................... 107
- Updating PSBs ......................................................... 108
- Adding or modifying a sparse exit routine ..................... 109
  - Single-character method ........................................ 110
  - User-written method ............................................. 110
  - Maintenance functions .......................................... 110
  - Coding requirements ............................................. 111
  - Applying maintenance ............................................ 113
- Adding or modifying a partitioned exit routine .............. 114
  - Vertical sequential processing mode ......................... 115
  - Lateral sequential processing mode .......................... 116
  - Maintenance functions .......................................... 116
  - Coding requirements ............................................. 116
  - Applying maintenance ............................................ 119
When you have defined your primary DEDB, you can perform the following types of maintenance:

- add another PFX index to a primary DEDB that already has a PFX index
- delete index registration from a primary DEDB
- update primary DBDs
- update PSBs
- add a sparse exit routine to control how many entries should be included in the PFX index
- add a partitioned exit routine that randomizes data sequentially into partitioned SHISAM or HISAM indexes

The maintenance procedures described in this section are stand-alone procedures. You can perform them any time after completing the original steps in the index registration process.

Each maintenance procedure described in this chapter requires you to update the ACBLIB and PFXLIB. If IMS is not active, you must restart it so that you can bring in the new control blocks. If IMS is active, you must perform an online ACB change and start all databases that were affected by the change.

**Online IMS maintenance**

When you update a primary DEDB in an IMS online environment, Fast Path Indexer/EP automatically maintains the associated PFX indexes. Fast Path Indexer/EP transparently ensures that the data and the hierarchical structure of the PFX index stays in synchronization with the primary DEDB.
If you modify a primary DBD that affects a PFX index, you must execute the PFXAGEN utility again to re-register the primary DBD to Fast Path Indexer/EP. This re-registration ensures that the Fast Path Indexer/EP control blocks synchronize with the online ACBs. Also, if the primary DBD change affects the segment on which the PFX index is based, you might need to

- re-register the PFX index to Fast Path Indexer/EP
- rebuild the PFX index before the application uses it

### Adding a PFX index

To add a PFX index to a primary DEDB, perform the following tasks:

1. Register the PFX index (page 99)
2. Build the PFX index (page 104)
3. Implement the PFX index in IMS (page 104)

### Registering the PFX index

To register the PFX index, perform the following steps:

1. Modify the primary DBD to include the following Fast Path Indexer/EP macros: PFXCHILD, PFXPART, PFXXDFLD, and PFXGEN. For more information about each macro, see Chapter 4, “Fast Path Indexer/EP PFX index macros.”

Fast Path Indexer/EP macros do not affect the IMS DBD control block. The macros create control sections (CSECTs) that are used exclusively by the Fast Path PFXAGEN utility. Figure 34 illustrates how to modify a primary DEDB to include Fast Path Indexer/EP macros.

#### Figure 34  Modifying a primary DEDB to include Fast Path Indexer/EP macros (part 1 of 2)

<table>
<thead>
<tr>
<th>DBD</th>
<th>NAME=DIVSSD,ACCESS=DEDB,RMNAME=(DIVSSD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREA DD1=DIVSSD1,DEVICE=3380,SIZE=4096,UOW=(15,05),</td>
<td></td>
</tr>
<tr>
<td>ROOT=(15,05)</td>
<td></td>
</tr>
<tr>
<td>SEGM NAME=...</td>
<td></td>
</tr>
<tr>
<td>SEGM NAME=...</td>
<td></td>
</tr>
<tr>
<td>SEGM NAME=INSTR,PARENT=(LOCN),BYTES=(56,56),TYPE=DIR</td>
<td></td>
</tr>
<tr>
<td>FIELD NAME=(INSTR#,SEQ,U),BYTES=5,START=3,TYPE=C</td>
<td></td>
</tr>
</tbody>
</table>
Registering the PFX index

1. Run IMS DBDGEN to create a new primary DBD that includes the required Fast Path Indexer/EP macros.

   Include the PFX macro library (PFPMAC or IMMAC distribution library) to the DBDGEN JCL SYSLIB concatenation. The Fast Path Indexer/EP macro statements used in IMS DBDGEN create a control section (CSECT) at the end of the primary DBD. The CSECT contains the information needed to define the primary DEDB to Fast Path Indexer/EP.

   The assembly output of the new DBD shows the recommended PFX index database definition parameters for each PFX index database, as shown in the example in Figure 35.

Figure 35  Recommended PFX index database source output by Fast Path Indexer/EP during primary DEDB DBDGEN (part 1 of 2)

| FIELD NAME=(INSTRSSN),BYTES=009,START=00038,TYPE=C |
| FIELD NAME=(INSTRNME),BYTES=030,START=00008,TYPE=C |
| FIELD NAME=(INSTRPHN),BYTES=010,START=00047,TYPE=C |
| FIELD NAME=(INSTAREA),BYTES=003,START=00047,TYPE=C |
| PFXCHILD NAME=((PFXROROT),DIVSI2), ACCESS=(SHISAM) |
| PFXXDFLD NAME=SRCVALU,SEGMENT=INSTR,SRCH=(INSTRSSN), DDATA=(INSTRNME,INSTRPHN),NULLVAL=X'F0' |

Figure 34  Modifying a primary DEDB to include Fast Path Indexer/EP macros (part 2 of 2)
3 Generate the PFX index DBD.

You can use the recommended PFX index database definition parameters produced in the output for step 2 to generate the PFX index DBD.

4 Run IMS DBDGEN for PFX indexes to create a new DBD for each PFX index.

For information about IMS DBDGEN, see the IBM *IMS/ESA Utilities Reference* manual.

5 Create or update the PSB source.

You can retrieve source data by using a PFX index, or you can retrieve the data directly from the primary DEDB. Both methods require that you define and generate PSBs.

**Retrieving data by using a Fast Path Indexer/EP PFX index (alternate processing sequence)**

If you intend to retrieve or update source data through the PFX index by using an alternate processing sequence, the primary DBDs that you register with Fast Path Indexer/EP must include the IMS PROCSEQ keyword on the PCB statement of the PSB. PROCSEQ generates a list of names and PCB offsets in the PSB. This list generates the appropriate data in the PSBMs, which are internal control blocks that tell Fast Path Indexer/EP how the PSB is configured in an IMS-dependent region.

---

**Figure 35  Recommended PFX index database source output by Fast Path Indexer/EP during primary DEDB DBDGEN (part 2 of 2)**

```plaintext
+*,*    ***********************************************
+*,*    DBD  NAME=DIVSI2,
+*,*    ACCESS=(SHISAM,VSAM)
+*,*    DATASET DD1=?????????,
+*,*    DEVICE=????,SIZE=????,RECORD=(69)
+*,*    SEGM  NAME=PFXROOT,
+*,*    PARENT=0,
+*,*    BYTES=69
+*,*    FIELD NAME=(SRCVALU,SEQ,U),
+*,*    START=1,
+*,*    BYTES=9
+*,*    DBDGEN
+*,*    FINISH
+*,*    END
+*,*    ***********************************************
```
Registering the PFX index

Fast Path Indexer/EP does not require that you use PROCSEQ on the PCB statement to indicate that the primary DEDB was registered to Fast Path Indexer/EP. You use PROCSEQ only to indicate that you are going to process the data through the PFX index by using an alternate processing sequence.

**NOTE**

IMS PSBGEN allows the PROCSEQ keyword to be included on the PCB statement, but IMS ACBGEN for version 11.1 and earlier does not support this keyword for DEDB PCBs. Because of this restriction, PFXAGEN updates the PSB to eliminate the PROCSEQ keyword before the IMS ACBGEN; otherwise, IMS ACBGEN returns an error code.

All segments down to the target must include the SENSEG (sensitive segment) statement from the root through the source segment. Each PCB must include at least one SENSEG statement.

Figure 36 shows a PSB that contains a Fast Path Indexer/EP-registered PCB. This PCB contains the PROCSEQ keyword and uses a PFX index to retrieve data from the primary DEDB. The target segment is Student, and the SENSEGs appear in their original order, not the inverted structure.

**Figure 36 PCB example for a primary DEDB with PROCSEQ**

<table>
<thead>
<tr>
<th>PCB</th>
<th>TYPE=DB,PROCSEQ=DIVSI3,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DBDNAME=DIVSSD,KEYLEN=60,PROCOPT=A</td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=COURSE,PARENT=0</td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=LOCN,PARENT=COURSE</td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=INSTR,PARENT=LOCN</td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=STUDENT,PARENT=LOCN</td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=PREREQ,PARENT=COURSE</td>
</tr>
<tr>
<td>PSBGEN</td>
<td>PSBNAME=APPLGM2,</td>
</tr>
<tr>
<td></td>
<td>CMPAT=YES,</td>
</tr>
<tr>
<td></td>
<td>LANG=COBOL</td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>

Retrieving data through the primary DEDB (primary processing sequence)

If you are retrieving data from the primary DEDB and you do not want the data retrieved in an index order, you must include a PCB for the primary DEDB that does not include a PROCSEQ keyword (Figure 37).

**Figure 37 PCB example for a primary DEDB without PROCSEQ (part 1 of 2)**

<table>
<thead>
<tr>
<th>PCB</th>
<th>TYPE=DB,DBDNAME=DIVSSD,KEYLEN=60,PROCOPT=G</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSEG</td>
<td>NAME=COURSE,PARENT=0</td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=LOCN,PARENT=COURSE</td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=INSTR,PARENT=LOCN</td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=STUDENT,PARENT=LOCN</td>
</tr>
<tr>
<td>SENSEG</td>
<td>NAME=PREREQ,PARENT=COURSE</td>
</tr>
</tbody>
</table>
Registering the PFX index

Chapter 6 Primary DEDB and PFX index control block maintenance

If you process a stand-alone index, you must specify a PCB for that index; do not include a PROCSEQ keyword (Figure 38).

Because Fast Path Indexer/EP uses date/time stamps to compare the data between Fast Path Indexer/EP registration control blocks and ACBs, you must use PFXAGEN to generate the related ACB control blocks.

6 Run IMS PSBGEN.

IMS PSBGEN creates at least one PSB that refers to the primary DEDB that you just registered to Fast Path Indexer/EP. If other PSBs are associated with a specific application that will use Fast Path Indexer/EP, you must also generate those PSBs.

7 Run PFXAGEN.

PFXAGEN invokes IMS ACBGEN to generate all ACBs; these ACBs do not have to be Fast Path Indexer/EP-related. PFXAGEN creates database registration entries that IMS-dependent regions use to identify databases that use Fast Path Indexer/EP PFX indexes.

NOTE

BMC recommends that you replace your current IMS ACBGEN procedure with the PFXAGEN procedure to avoid out-of-sync situations that could occur between control blocks residing in PFXLIB and ACBs residing in ACBLIB.

Fast Path Indexer/EP provides a front-end process to the IMS ACBGEN process. Fast Path Indexer/EP does not perform its own ACBGEN; it always uses IMS ACBGEN.

You can include a PSBSRC DD statement so that PFXAGEN can create PSB source for the modified PSBs. PFXAGEN assembles and links the modified PSBs. It generates an ACB for all requested primary DBDs, PFX index DBDs, and PSBs. PFXAGEN reprocesses the PSBMs and includes the date/time stamp from the ACB so that Fast Path Indexer/EP can maintain a relationship between the PSBMs and the ACBs.

For more information, see Chapter 5, “Fast Path Indexer/EP PFXAGEN utility.”
Building the PFX index

Execute the BUILD command provided by Fast Path Indexer/EP to extract data from the primary DEDB and build the PFX index.

Input to the BUILD command can be either the area data sets or XSCAN-created data sets. You can use the BUILD command to customize the index build process, including specific areas within the primary DEDB as well as the PFX index itself. For more information, see Chapter 9, “Index build utility.”

Implementing the PFX index in IMS

You implement the PFX index in IMS by refreshing the control blocks in the control region. To refresh the control blocks, update the IMS STAGE1 with the new PFX index database definition, and then execute the IMS generation process. The procedure that follows depends on whether IMS is active, IMS is inactive, or you are using DELTA IMS.

To refresh control blocks if IMS is active

1 Copy the staging MODBLKS data set to the INACTIVE online MODBLKS data set and then perform a MODBLKS online change.

2 Copy the staging ACBLIB and PFXLIB (updated in step 7 on page 103) to the INACTIVE online ACBLIBx/PFXLIBx pair and then perform an ACB online change.

   Fast Path Indexer/EP automatically uses the PFXLIB that matches the active ACBLIB.

To refresh control blocks if IMS is inactive

1 Copy the staging MODBLKS data set to MODBLKSA and MODBLKSB data set.

2 Copy the staging ACBLIB and PFXLIB (updated in step 7 on page 103) to ACBLIBA/PFXLIBA and ACBLIBB/PFXLIBB.

3 Restart IMS.
To refresh control blocks if you use the BMC DELTA IMS product for Fast Path Indexer/EP databases

1. Execute the ADD command for the PFX index database definition.

2. Copy the staging ACBLIB and PFXLIB (updated in step 7 on page 103) to the ACTIVE online ACBLIBX/PFXLIBx pair.

3. Execute the RELOAD command for the Fast Path Indexer/EP DEDB, including sensitive PSBs.

**NOTE**
If you use the BMC DELTA IMS product for Fast Path Indexer/EP databases, you must always list the Fast Path Indexer/EP PFX index before the Fast Path Indexer/EP primary DEDB when executing ADD, REVISE, and RELOAD commands.

**Deleting PFX index registration**

If you no longer want certain PFX indexes associated with your primary DEDB, you must remove the associated index registrations from your primary DEDB.

**To delete PFX index registration**

1. Modify the primary DBD to remove the PFX macros:
   - If you want to delete only selected PFX indexes associated with a primary DEDB, delete the PFXCHILD and PFXXDFLD macro statements from the primary DEDB for each PFX index registration to be deleted.
   - If you want to delete all PFX indexes associated with a primary DEDB, delete every occurrence of PFXCHILD, PFXXDFLD, and PFXGEN macro statements from your primary DBD.

2. Run IMS DBDGEN to generate the updated primary DBD in the DBDLIB.

3. Update the PSB source.

If any PSBs include PCBs for the primary DEDB that contain the PROCSEQ keyword, you must remove PROCSEQ. Fast Path Indexer/EP supports the PROCSEQ keyword for DEDBs; IMS does not.

4. Run IMS PSBGEN to update the PSBs in the PSBLIB.

5. Run PFXAGEN to update the ACBLIB and the PFXLIB.
6 Refresh the control blocks in the control region.

Use your standard migration procedures to stage your new ACBLIB and PFXLIB to the active/inactive ACBLIBx/PFXLIBx libraries:

- If you are staging ACBs to ACBLIBA, stage PFXLIB to PFXLIBA.
- If you are staging ACBs to ACBLIBB, stage PFXLIB to PFXLIBB.

7 When you complete your migration procedures, determine whether IMS is active:

- If IMS is active, you can perform an ACB online change to refresh the active ACBLIB. Fast Path Indexer/EP automatically uses the active PFXLIBx that matches ACBLIBx.
- If IMS is not active, issue a START command and bring in the new control blocks.

## Updating primary DBDs

Depending on the type of updates that you apply to a primary DBD, the updates might or might not affect the associated PFX index. This section describes the steps that you must complete to update primary DBDs.

### Updating DBDs that affect PFX indexes

The following updates to the primary DBD affect the associated PFX indexes:

- changing the hierarchy of your primary DEDB
- changing a segment in your primary DEDB that affects a PFX index

When you apply maintenance to primary DBDs that affect associated PFX indexes, you follow the standard registration procedure. For more information about the steps included in this section, see Chapter 3, “Adding index functionality to the Fast Path environment.”

**To update DBDs that affect PFX indexes**

1 Modify the primary DBD.

2 Run IMS DBDGEN to generate the updated primary DBD in the DBDLIB.

3 Using the recommended primary DBD from step 2, generate the PFX index DBD.
4 Run IMS DBDGEN for PFX indexes.

5 (optional) Update the PSB source, and run IMS PSBGEN to update the PSBs in the PSBLIB.

If you complete this step, include PROCSEQ on the appropriate PCBs.

6 Run PFXAGEN.

7 Execute the BUILD command provided by Fast Path Indexer/EP to extract data from the primary DEDB and build the PFX index.

Input to the BUILD command can be either the area data sets or XSCAN-created data sets. You can use the BUILD command to customize the index build process, including specific areas within the primary DEDB and the PFX index itself. For more information, see Chapter 9, “Index build utility.”

8 Refresh the control blocks in the control region.

Use your standard migration procedures to stage your new ACBLIB and PFXLIB to the active/inactive ACBLIBx/PFXLIBx libraries:

- If you are staging ACBs to ACBLIBA, stage PFXLIB to PFXLIBA.
- If you are staging ACBs to ACBLIBB, stage PFXLIB to PFXLIBB.

9 When you complete your migration procedures, determine whether IMS is active:

- If IMS is active, you can perform an ACB online change to refresh the active ACBLIB. PFX automatically uses the active PFXLIBx that matches ACBLIBx.
- If IMS is not active, issue a START command and bring in the new control blocks.

**Updating primary DBDs only**

If you modify the primary DBD and it does not affect the associated PFX indexes, perform the following steps before using Fast Path Indexer/EP again:

1 Modify the primary DBD.

2 Run IMS DBDGEN to generate the updated primary DBD in the DBDLIB.

3 (optional) Update the PSB source, and run IMS PSBGEN to update the PSBs in the PSBLIB.
Updating PSBs

Some situations require you to update PSBs. For example, if you change a processing option or add the PROCSEQ keyword for a PFX index, you must update the appropriate PSB to reflect this change.

To update PSBs

1 Update the PSB source.

To update a PSB, use the original PSB source. You must include PROCSEQ on all PCBs that use the alternate processing sequence.

2 Run IMS PSBGEN to update the PSBs in the PSBLIB.

3 Run PFXAGEN to update the ACBLIB and the PFXLIB.

4 Refresh the control blocks in the control region.

Use your standard migration procedures to stage your new ACBLIB and PFXLIB to the active/inactive ACBLIBx/PFXLIBx libraries:

- If you are staging ACBs to ACBLIBA, stage PFXLIB to PFXLIBA.
- If you are staging ACBs to ACBLIBB, stage PFXLIB to PFXLIBB.

5 When you complete your migration procedures, determine whether IMS is active:

- If IMS is active, you can perform an ACB online change to refresh the active ACBLIB. PFX automatically uses the active PFXLIBx that matches ACBLIBx.
- If IMS is not active, issue a START command and bring in the new control blocks.
When you complete your migration procedures, determine whether IMS is active:

- If IMS is active, you can perform an ACB online change to refresh the active ACBLIB. PFX automatically uses the active PFXLIBx that matches ACBLIBx.
- If IMS is not active, issue a START command and bring in the new control blocks.

Adding or modifying a sparse exit routine

Fast Path Indexer/EP supports the creation of a sparse exit routine to exclude selected records from a PFX index. If the routine specifies that a particular record should be suppressed, a pointer record is not placed in the PFX index.

This feature is useful because you might not always want to include a pointer from the PFX index to every record in the primary DEDB. For example, you might want to index fields that sometimes contain blanks or zeros. Because these values have no significant meaning, you can code a sparse exit routine to exclude them.

BMC does not recommend coding an exit routine that multiple primary DEDBs share. Sharing an exit routine requires cycling IMS to load a new (changed) copy of a shared exit routine. Therefore, using IMS Online Change to reload an ACB for any primary DEDB that shares the exit will reload the original, unchanged exit routine. In contrast, if the exit routine is not shared, IMS Online Change will load the new (changed) exit when the primary ACB is reloaded.

Fast Path Indexer/EP uses the sparse exit routine in the same way that IMS uses a PFX index database maintenance exit routine. If you need to add an exit routine to Fast Path Indexer/EP, follow the same coding requirements described in the IBM IMS EXIT Routine Reference manual.

Fast Path Indexer/EP provides the following methods for coding a sparse exit routine:

- using a single-character value comparison between the NULLVAL character and the pointer segment value
- calling a user-written exit to determine whether the pointer segment should be suppressed
Single-character method

The first type of PFX index sparse maintenance exit routine uses the NULLVAL keyword during the registration process. The value associated to NULLVAL is a single character; for example: NULLVAL=C’. Fast Path Indexer/EP compares this single-character value to the portion of the source segment that will be used as the index key value. When the product performs a comparison between the single-character value and the single-source or multiple-source segment value, it looks for a match between the two values. If a match exists, Fast Path Indexer/EP excludes the pointer segment from the PFX index.

User-written method

A single-character comparison might not be sufficient to exclude all unwanted pointer segments from the PFX index. The second type of sparse exit routine lets you write your own user exit. This exit examines all pointer segments in the primary DE DB and dynamically excludes specific pointer segments from the PFX index.

You specify this user-written exit during the index registration process by coding the EXTRTN keyword on the PFXXDFLD macro. For more information about the EXTRTN keyword, see Table 10 on page 67.

Maintenance functions

The sparse exit routine is called during the index maintenance for the functions listed in Table 12.

<table>
<thead>
<tr>
<th>Function</th>
<th>When the exit routine is called</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert</td>
<td>after the index segment is built but before it is inserted into the PFX index database</td>
</tr>
<tr>
<td>replace</td>
<td>before retrieving the old segment, and before replacing or inserting the new segment</td>
</tr>
<tr>
<td></td>
<td>Note: The first call enables the exit to determine whether the segment should be on the PFX index database. If the exit indicates that the segment should not be on the PFX index, no GET processing is performed on the PFX index.</td>
</tr>
<tr>
<td>delete</td>
<td>after an index segment is built</td>
</tr>
<tr>
<td></td>
<td>This call determines whether the segment should be on the PFX index database. If the exit indicates that the segment should not be part of the PFX index database, no DELETE processing is performed on the PFX index.</td>
</tr>
</tbody>
</table>
Coding requirements

When you code a sparse exit routine for Fast Path Indexer/EP, you must follow the same coding conventions that IBM requires. These conventions are detailed in the IBM *IMS/ESA Customization Guide: Database* manual.

For information about a secondary index database maintenance exit routine, see the IBM *IMS/ESA Database Administration Guide* manual.

You must comply with the following requirements to code a sparse exit routine:

- Ignore the first parameter of the sparse index in batch. If you run a sparse exit routine with any of the Fast Path Indexer/EP index commands (BUILD, VERIFY or RESYNC), the first keyword (PST) of the sparse index is not available; therefore, PST should not be used.

- You must use AMODE=31 and RMODE=ANY when you add an index sparse maintenance exit routine.

- Use reentrant and reusable link attributes. Because you load the sparse exit routine into IMS common service area (CSA) or extended CSA (ECSA), you must link-edit the routine with reentrant (RENT) and reusable (REUS) attributes.

- Use register settings at entry to exit. Fast Path Indexer/EP uses standard linkage conventions with pre-chained save areas. At entry to the exit, the registers are set as shown in Table 13.

Table 13 Registers at entry to user-written exit

<table>
<thead>
<tr>
<th>Register</th>
<th>Setting</th>
</tr>
</thead>
</table>
| 1        | IMS PST address  
The PST address is not available in batch build. |
| 2        | index segment address |
| 3        | index maintenance exit parameter list address |
| 4        | index source segment address |
| 13       | pre-chained save area address |
| 14       | return address |
| 15       | entry address  
See Table 14 for the parameters passed to this register. |
The parameters listed in Table 14 are passed in register 3 to the sparse exit routine.

Table 14  Parameter list for user-written exit

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length</th>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>8</td>
<td>Target segment name</td>
</tr>
<tr>
<td>+8</td>
<td>8</td>
<td>PFXXDFLD name</td>
</tr>
<tr>
<td>+16</td>
<td>8</td>
<td>Exit routine name</td>
</tr>
<tr>
<td>+24</td>
<td>4</td>
<td>Exit entry point address</td>
</tr>
<tr>
<td>+28</td>
<td>2</td>
<td>Exit parameter length</td>
</tr>
</tbody>
</table>

Use register settings at time of return. All registers should be returned as they were at entry (except register 15). Register 15 should contain the values listed in Table 15.

Table 15  Register 15 at time of return from user-written exit

<table>
<thead>
<tr>
<th>Register 15</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal to 0</td>
<td>Continue to index this segment.</td>
</tr>
<tr>
<td>Not equal to 0</td>
<td>Do not index this segment.</td>
</tr>
</tbody>
</table>

Use the index segment format. The index segment entry is built in PFX internal format for SHISAM or HISAM indexes, and has the format shown in Figure 39 or Figure 40.

Figure 39  Structure for a standard PFX index format for user-written exit
Applying maintenance

If you add or modify a sparse exit routine, perform the following steps before using Fast Path Indexer/EP again:

1. Code the sparse exit routine.
   
   For more information, see “Coding requirements” on page 111.

2. Modify the primary DBD.
   
   To add the sparse exit routine, include the EXTRTN keyword on the PFXXDFLD macro statement. This macro statement resides on the primary DBD. For more information about the EXTRTN keyword, see Table 10 on page 67.

3. Run IMS DBDGEN to update the primary DBD to the DBDLIB.

4. Run PFXAGEN to update the ACBLIB and the PFXLIB.

   For more information, see “Generating ACBs” on page 78.
5 Complete this step if the primary DEDB is offline (but skip to step 6 if the DEDB is online):

A Execute the BUILD command provided by the Fast Path Indexer/EP product to extract data from the primary DEDB and build the PFX index.

Input to the BUILD command can be either the area data sets or XSCAN-created data sets. You can use the BUILD command to customize the index build process, including specific areas within the primary DEDB as well as the PFX index itself. For more information, see Chapter 9, “Index build utility.”

B Refresh the control blocks in the control region (see step 8).

6 Complete this step if the primary DEDB is online:

A Refresh the control blocks in the control region (see step 8).

B Execute the RESYNC command provided by the Fast Path Indexer/EP product to resynchronize the PFX index with its associated primary DEDB. For more information, see Chapter 10, “Index validation utilities.”

7 Refresh the control blocks in the control region.

Use your standard migration procedures to stage your new ACBLIB and PFXLIB to the active/inactive ACBLIBx/PFXLIBx libraries:

- If you are staging ACBs to ACBLIBA, stage PFXLIB to PFXLIBA.
- If you are staging ACBs to ACBLIBB, stage PFXLIB to PFXLIBB.

8 When you complete your migration procedures, determine whether IMS is active:

- If IMS is active, you can perform an ACB online change to refresh the active ACBLIB. PFX automatically uses the active PFXLIBx that matches ACBLIBx.
- If IMS is not active, issue a START command and bring in the new control blocks.

---

**Adding or modifying a partitioned exit routine**

The partitioned exit routine controls the placement of data in a partitioned SHISAM or HISAM index. The partitioned exit routine sequentially randomizes data into a partitioned SHISAM or HISAM index. By partitioning the PFX index, you eliminate the maximum 4-GB size restriction imposed by IMS since each partition is a physically separate DBD and KSDS data set.
You can specify lateral sequential processing mode or vertical sequential (default) processing mode to be used in conjunction with a partitioned exit routine. Fast Path Indexer/EP must be able to control the range of sequential key values by partition. To accomplish this control, Fast Path Indexer/EP uses the partitioned exit routine. Figure 41 illustrates both types of processing modes for a partitioned SHISAM or HISAM index. The difference between vertical and lateral is the location in the reading sequence where IMS returns the GB status code to indicate that there is no more data to process. For more information about vertical and lateral processing modes, see page 115.

**Figure 41  Partitioned exit routine processing modes**

**Vertical sequential processing mode**

Use the default vertical sequential processing mode to distribute a primary DEDB area separately from the other areas.

Vertical sequential processing mode distributes data down each partition and does not process beyond a certain key range. When this process reaches the end of a partition, IMS issues a GB status code indicating that there is no more data to process.
Lateral sequential processing mode

Using Figure 41 as an example, vertical sequential mode could distribute data on a one-to-one basis from Area 1 of the primary DEDB to Partition 1 of the SHISAM or HISAM index. Vertical sequential mode could distribute data the same way for Area 2 and Partition 2 and for Area 3 and Partition 3.

Vertical sequential mode does not necessarily have to distribute data on a one-to-one basis. Vertical sequential mode could distribute data from Area 1 of the primary DEDB to Partition 3 of the SHISAM or HISAM index. Vertical sequential mode always distributes data from one area of the primary DEDB to one partition of the SHISAM or HISAM index partition.

Lateral sequential processing mode

Use the lateral sequential processing mode to distribute all DEDB source areas at one time.

Lateral sequential mode distributes data across each partition. When this process reaches what would normally be the end of a partition, Fast Path Indexer/EP embeds a blank status code and continues the process. When lateral sequential mode reaches the end of all key ranges in the partition, IMS issues a GB status code.

Using Figure 41 as an example, a lateral sequential mode could distribute data from Area 1 of the primary DEDB to Partition 1 of the SHISAM or HISAM index. Lateral sequential mode would distribute data until it reaches the end of the specified key range (the end of Partition 3).

Maintenance functions

The partitioned exit routine is called during the index maintenance for the functions listed in Table 12.

Coding requirements


You must comply with the following requirements to code a partitioned exit routine:

- You must use AMODE=31 and RMODE=ANY when you use a partitioned exit routine.
Use reentrant and reusable link attributes. Because you load the partitioned exit routine into IMS CSA or ECSA, you must link-edit the routine with reentrant (RENT) and reusable (REUS) attributes.

When you code the partition exit, you must return a pointer in Register 15 for a valid partition. Otherwise, the product issues message BMC123172E and abends with user abend U3847.

At entry to the exit routine, the registers are set as shown in Table 16.

Table 16  Registers at entry to partitioned exit routine

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Address of PARM list</td>
</tr>
<tr>
<td>R2</td>
<td>Address of index segment key</td>
</tr>
<tr>
<td>R4</td>
<td>Address of 32-byte work area</td>
</tr>
<tr>
<td>R5</td>
<td>Address of start of index list</td>
</tr>
</tbody>
</table>

Figure 42 shows an example of how to code a partitioned exit routine. Two PFX indexes are defined with partitions. DIVSI4 has 10 partitions, and DIVSI7 has 3; the code specifies the key ranges that each partition will contain.
Figure 42  Partitioned exit routine example (part 2 of 3)

```
CLC  0(8,R5),=CL8'DIVSI4' INDEX 4?
BNE  PART100  NO - MUST BE DIVSI7
LA  R7,DIVSI4  GET DIVSI4 TABLE
B  PART200  START TABLE SCAN
PART100  DS  OH
LA  R7,DIVSI7  GET DIVSI7 TABLE
************************************************************************
*     GET PARMS PASSED                                                 *
************************************************************************
PART200  DS  OH
SR  R1,R1
IC  R1,PARTKEYL  GET KEY LENGTH
L  R3,PART#SRC  GET DIMINITIONS OF TABLE
LR  R15,R5  START WITH FIRST
************************************************************************
*     SCAN TABLE FOR VALUE                                             *
************************************************************************
PART300  DS  OH
CLC1  CLC  0(*-*),R7),0(R2) KEY FOUND?
EX  R1,CLC1  FOUND KEY
BNL  EXIT  YES
LA  R8,1(R1)  LENGTH OF KEY
SLL  R8,1  TIMES 2
LA  R7,0(R7,R8)  BUMP TO NEXT VALUE
LA  R15,8(R15)  POINT TO NEXT INDEX
BCT  R3,PART300  DO NEXT VALUE
ABEND 1000,DUMP  KEY VALUE NOT FOUND
EXIT  DS  OH
L  R13,4(R13)
ST  R15,16(R13)
LM  R14,R12,12(R13)
BR  R14
************************************************************************
*     KEY RANDOMIZING TABLES                                           *
************************************************************************
DIVSI4  DS  OF
DC  C'099','C'000'  HIGH ,LOW VALUE PARTITION 1
DC  C'199','C'100'  HIGH ,LOW VALUE PARTITION 2
DC  C'299','C'200'  HIGH ,LOW VALUE PARTITION 3
DC  C'399','C'300'  HIGH ,LOW VALUE PARTITION 4
DC  C'499','C'400'  HIGH ,LOW VALUE PARTITION 5
DC  C'599','C'500'  HIGH ,LOW VALUE PARTITION 6
DC  C'699','C'600'  HIGH ,LOW VALUE PARTITION 7
DC  C'799','C'700'  HIGH ,LOW VALUE PARTITION 8
DC  C'899','C'800'  HIGH ,LOW VALUE PARTITION 9
DC  C'999','C'900'  HIGH ,LOW VALUE PARTITION 10
*
DIVSI7  DS  OF
DC  CL4'000G'  HIGH KEY VALUE - PART 1
```
Applying maintenance

If you add or modify a partitioned exit routine to your PFX index database, perform the following steps before using the PFX index again:

1. Code the partitioned exit routine.
   
   For more information, see “Coding requirements” on page 116.

2. Modify the primary DBD.
   
   To add the partitioned exit routine, include the PARTEXIT keyword on the PFXXDFLD macro statement. This macro statement resides on the primary DBD. For more information about the PARTEXIT keyword, see Table 10 on page 67.

3. Run IMS DBDGEN to replace the primary DBD in the DBDLIB.

4. Generate the PFX index DBD.

   Use the recommended primary DBD that was generated during step 2 for Fast Path Indexer/EP PFX index DBDs.

### Figure 42  Partitioned exit routine example (part 3 of 3)

```plaintext
DC   CL4'000A'     LOW KEY VALUE - PART 1
*  
DC   CL4'000R'     HIGH KEY VALUE - PART 2
DC   CL4'000H'     LOW KEY VALUE - PART 2
*  
DC   CL4'999Z'     HIGH KEY VALUE - PART 3
DC   CL4'000S'     LOW KEY VALUE - PART 3
*  
**************************************************************************
*  D S E C T S                                                      *
**************************************************************************
PARMS   DSECT
**************************************************************************
**   PFX PARTITIONED INDEX EXIT INTERFACE BLOCK                      **
**************************************************************************
PARTDBDN DC   CL8' '     SOURCE DATA BASE NAME
PARTSGNM DC   CL8' '     TARGET SEGMENT NAME
PARTFLDN DC   CL8' '     INDEXED (XDFLD) NAME
PARTNAME DC   CL8' '     EXIT ROUTINE NAME
PART#SRC DC   A(0)       NUMBER OF INDEXES FOR THIS SEG
PARTKEYL DC   AL1(0)     EXEC LENGTH OF INDEX KEY
END
```
Applying maintenance

5 Run IMS DBDGEN for PFX indexes.

6 Run PFXAGEN to update the ACBLIB and the PFXLIB.

7 Complete this step if the primary DEDB is offline (but skip to step 8 if the DEDB is online):
   A Execute the BUILD command provided by the Fast Path Indexer/EP product to extract data from the primary DEDB and build the PFX index.
   
   Input to the BUILD command can be either the area data sets or XSCAN-created data sets. You can use the BUILD command to customize the index build process, including specific areas within the primary DEDB as well as the PFX index itself. For more information, see Chapter 9, “Index build utility.”

   B Refresh the control blocks in the control region.
   
   Use your standard migration procedures to stage your new ACBLIB and PFXLIB to the active/inactive ACBLIBx/PFXLIBx libraries.

   C When you complete your migration procedures, issue a START command and bring in the new control blocks.

8 Complete this step if the primary DEDB is online:

   A Refresh the control blocks in the control region.

   B Execute the Online Change command to switch active ACBLIBx and PFXLIBx libraries.

   If you are staging ACBs to ACBLIBA, you must stage PFXLIB to PFXLIBA. If you are staging ACBs to ACBLIBB, you must stage PFXLIB to PFXLIBB.

   C Execute the RESYNC command provided by the Fast Path Indexer/EP product to resynchronize the PFX index with its associated primary DEDB. For more information, see Chapter 10, “Index validation utilities.”
Utilities for PFX and IBM native indexes

This part presents the following topics:

Chapter 7
   Index utility commands and processing modes ........................................ 123

Chapter 8
   Index scan utility ......................................................................................... 141

Chapter 9
   Index build utility ......................................................................................... 153

Chapter 10
   Index validation utilities ................................................................................ 173

Chapter 11
   Index registration and verification utility ................................................... 193
Chapter 7 Index utility commands and processing modes

This chapter introduces command-driven utilities that are available in the Fast Path Indexer/EP product. It also describes the requirements and performance considerations for executing these commands in offline mode or BMP (online) mode.

**NOTE**

All topics in this chapter apply to both PFX indexes and IBM native indexes, except where noted.

This chapter discusses the following topics:

- Fast Path Indexer/EP command set .................................................. 124
  - XSCAN command ................................................................. 124
  - BUILD command ............................................................... 124
  - VERIFY command .............................................................. 125
  - RESYNC command ............................................................. 125
- Processing modes ................................................................. 126
  - Offline processing ............................................................. 126
    - Execution JCL for offline processing .................................. 127
    - Region size ................................................................. 127
    - STEPLIB requirements .................................................... 127
    - Identifying the functions to be performed ......................... 128
    - Identifying the DEDB areas to be processed ..................... 128
    - Dynamic allocation of data sets ...................................... 128
    - Use of DBRC ............................................................... 130
    - Caching in offline mode ............................................... 132
  - Online (BMP) processing .................................................. 133
    - Execution JCL for BMP processing ................................. 134
    - Region size ............................................................... 134
    - STEPLIB requirements .................................................... 134
    - Identifying the functions to be performed ....................... 135
    - Creating the PSB for BMP processing ............................. 135
Fast Path Indexer/EP command set

The Fast Path Indexer/EP command set includes four primary commands. You can use them to build indexes and perform remediation processes (for example, to rebuild, synchronize, or ensure synchronization of the secondary index database with its associated primary DEDB). Each command can be executed as SYSIN input to the PFPMAIN program provided with all products in the Fast Path/EP Series. All commands that are introduced in this section are discussed in detail in subsequent chapters of this book.

XSCAN command

By using the Fast Path Indexer/EP index XSCAN function, you can scan a Fast Path primary DEDB in offline or online mode. Fast Path Indexer/EP provides the XSCAN command, the IX subcommand, and the associated keywords. The XSCAN command (when used with the IX subcommand and associated keywords) lets you scan a single area or multiple areas in a primary DEDB to locate index source segments for a defined index and then produce an output data set containing index information. This output data set can be used for input into the build, verify, and resynchronize utilities. For more information, see Chapter 8, “Index scan utility.”

BUILD command

By using the Fast Path Indexer/EP index build function, you can build individual or multiple indexes to a primary DEDB. You can build indexes by reading the primary DEDB, or by using an XSCAN data set that is created by the XSCAN command. The two approaches for building indexes are mutually exclusive. In addition, you can rebuild indexes after performing maintenance on the primary DEDB.
Fast Path Indexer/EP provides the BUILD command, the IX subcommand, and associated keywords. You can use them in various combinations to build secondary index databases to a primary DEDB. When you run a BUILD command, Fast Path Indexer/EP extracts data from the primary DEDB, sorts the data, optionally reads the XSCAN file, and populates the secondary index database.

The BUILD command can be executed in offline mode only. For more information, see Chapter 9, “Index build utility”

**VERIFY command**

By using the Fast Path Indexer/EP index verify function, you can verify the contents of a secondary index database to a primary DEDB in offline or online mode. Fast Path Indexer/EP provides the VERIFY command, the IX subcommand, and associated keywords. The VERIFY command enables you to diagnose the cause of an out-of-sync condition. For example, if the primary DEDB is recovered but the index is not recovered, you can use the VERIFY command to verify the contents of the secondary index database.

In addition, an XSCAN data set (created by the XSCAN utility) can be used in place of the DEDB area data set as input into the verify function; only one of these data set types (XSCAN data set or DEDB area data set) can be used as input. For more information, see Chapter 8, “Index scan utility” and Chapter 10, “Index validation utilities.”

**RESYNC command**

By using the Fast Path Indexer/EP index resynchronize function, you can resynchronize an index to a primary DEDB without taking the DEDB offline. Fast Path Indexer/EP provides the RESYNC command, the IX subcommand, and associated keywords. Unlike other online commands in the Fast Path/EP Series product set, the RESYNC command runs in an IMS batch message processing (BMP) region and can be executed only while the database is online (RESYNC cannot be executed offline).

In addition, an XSCAN data set (created by the XSCAN utility) can be used in place of the DEDB area data set as input into the resynchronize function; only one of these data set types (XSCAN data set or DEDB area data set) can be used as input. For more information, see Chapter 8, “Index scan utility” and Chapter 10, “Index validation utilities.”
Processing modes

Each of the command-driven functions provided by the Fast Path/EP Series products can operate in the processing modes that are listed in Table 17. Each processing mode is described in detail on the following pages.

Table 17  Available operational modes by Fast Path Indexer/EP command

<table>
<thead>
<tr>
<th>Command</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline</td>
<td>BMP</td>
</tr>
<tr>
<td>BUILD</td>
<td>X</td>
</tr>
<tr>
<td>RESYNC</td>
<td>X</td>
</tr>
<tr>
<td>VERIFY</td>
<td>X</td>
</tr>
<tr>
<td>XSCAN</td>
<td>X</td>
</tr>
</tbody>
</table>

System resource and performance considerations for the execution of the Fast Path Indexer/EP utility functions are dependent on the processing mode, the characteristics of the area being processed, and the optional processing that is requested by using command and subcommand keywords.

Offline processing

You can execute the primary commands that are associated with the Fast Path Indexer/EP build, verify, and XSCAN functions (BUILD, VERIFY, and XSCAN) in offline mode. Offline mode processes a primary DEDB or area that is offline to all IMS control regions. In offline mode, no other utility or process can be executed concurrently by using the database or area that is being processed by Fast Path Indexer/EP.

You can also execute the XSCAN command in IFP mode (see “Online (IFP) processing” on page 136) and the VERIFY command in BMP mode (see “Online (BMP) processing” on page 133).
Execution JCL for offline processing

Figure 43 illustrates the basic JCL needed to execute a Fast Path Indexer/EP utility function in offline mode. Dynamic allocation of the area data sets, PFXLIB (only needed for PFX indexes), and the ACB library is assumed.

Figure 43  Basic Fast Path/EP Series product JCL for offline processing

```plaintext
//PFP    EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
//         DD DISP=SHR,DSN=IMSVS.RESLIB
//PFPSYSIN DD *
-- BUILD command to identify primary DEDB--
-- IX subcommands to identify index databases--
/*
```

Region size

To ensure optimum performance, BMC recommends that you request the maximum available storage for execution of the Fast Path/EP Series product functions by specifying REGION=0M on the EXEC statement for the JOB step. Some sites might have different requirements. Check with your z/OS systems programmer to determine how to request the maximum REGION size.

STEPLIB requirements

Fast Path Indexer/EP utility functions must execute as an APF-authorized program. The product installation library (named BMC.PFP.LOAD in all examples) must be identified as an APF-authorized library. All of the data sets listed within the STEPLIB concatenation must also be APF authorized.

BMC recommends that the product installation library be placed first within the STEPLIB concatenation.

If other libraries precede the product installation library, you must ensure that none of these libraries contain a member name duplicating a member within the product library to ensure proper execution. The exception is member DFSRRC00, found within the PFP product library, which is not used in offline mode. For more information, see the Installation System User Guide and the Database Products for IMS Configuration Guide.
Identifying the functions to be performed

The PFPSYSIN control statements identify the functions to be performed. As shown in Figure 44, the DBD keyword is required for each command. The DBD keyword identifies the DEDB (DBD name) to be processed by the command. Multiple DEDBs can be processed within a single job step. All DBDs must be referenced by the same PSB for the VERIFY and RESYNC commands.

Figure 44   Specifying multiple function commands

| VERIFY DBD=dbdname1  |
| IX  |
| END |
| RESYNC DBD=dbdname2  |
| IX  |
| END |

Identifying the DEDB areas to be processed

The IAREA keyword identifies the names of the area or areas for which the function will build or verify an index. An example is shown in Figure 45 for the BUILD command. You can specify one or more area names, in any order, to be processed by the command. If you omit the IAREA keyword, the function processes all areas that are defined as part of the database.

Figure 45   Identifying the DEDB area for an index build

| BUILD DBD=dbdname1, IAREA=areaname |

Dynamic allocation of data sets

The minimum JCL that is required for execution of the Fast Path/EP Series product functions in offline mode is simple because many of the necessary data sets can be allocated dynamically. If you supply the appropriate DD statement in your JCL, the specified data sets are used. BMC recommends that you omit these DD statements so that dynamic allocation is used.
IMS ACB library

Fast Path Indexer/EP utility functions executing in offline mode require access to the IMS ACB library containing the database definitions (DMB) for the DEDB area or areas to be processed. If the IMSACB DD statement is omitted from the JCL, Fast Path Indexer/EP attempts to dynamically allocate it. The following IMSACB DD statements are used to identify the active ACB library:

- IMSACB
- IMSACBA
- IMSACBB
- OLCSTAT
- MODSTAT
- MODSTAT2
- PFXLIB
- PFXLIBA
- PFXLIBB

**NOTE**
The PFXLIB, PFXLIBA, and PFXLIBB DD statements are necessary for offline functions that process PFX indexes.

For details about using DD statements to identify the IMS ACB library, see the IMS/ESA System Administration Guide.

**NOTE**
The IMS/ESA release level of the ACB library must be the same as that of the RESLIB that is included in the STEPLIB DD statement concatenation.

Area data set

Fast Path Indexer/EP functions executing in offline mode require access to the area data set to be processed. If the areaname DD statement is omitted from the JCL, Fast Path Indexer/EP attempts to dynamically allocate it. The areaname DD statement identifies the area data set to be processed by Fast Path Indexer/EP.

The areaname DD statement is the area data set used as input to the index build or verify function. The IMSACB DD statement identifies the ACB library containing the database definition that describes the area referenced by the areaname DD statement.

If DBRC is active, the area is registered with DBRC, and the areaname DD statement refers to an area data set, then the areaname DD statement data set must match the registered area data set name.
If you are using dynamic allocation, do not include the areaname DD statement. Fast Path Indexer/EP attempts to obtain the data set name for allocation in the following order:

1. If the INPUT_DSN_MASK keyword is specified, it is used to generate the data set name for the build and verify functions.

2. If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.

3. The STEPLIB is searched for the DFSMDA member that contains the data set name for this area.

For the build and the verify function, the disposition of the dynamically allocated source area data set is DISP=SHR. The disposition for the secondary index database is DISP=OLD.

### Use of DBRC

When executing in offline mode, the Fast Path/EP Series products can interface with DBRC to perform several activities. The use of DBRC is optional but recommended. Activation and use of DBRC depends on how your IMS system has been defined and on the value set by the DBRC keyword of the GLOBAL command.

By default, Fast Path Indexer/EP uses the value that is set by the IMS system definition:

- If your IMS system has been defined to not use DBRC by default (the IMMSGEN is defined by using DBRC=(YES,NO)), this default is passed to the Fast Path/EP Series products, and DBRC is not activated during function processing. To activate DBRC, specify DBRC=YES on the Fast Path/EP GLOBAL command.

- If your IMS system has been defined to use DBRC by default (the IMMSGEN is defined by using DBRC=(YES,YES)), this default is passed to the Fast Path Indexer/EP, and DBRC is activated during function processing. To bypass DBRC processing, specify DBRC=NO on the GLOBAL command.

- If your IMS system has been defined to force the use of DBRC (the IMMSGEN is defined by using DBRC=FORCE), the Fast Path/EP Series products will activate DBRC during function processing. You cannot deactivate DBRC by specifying DBRC=NO on the GLOBAL command. If it is necessary to deactivate DBRC, you can specify DBRC=FORCEOFF on the GLOBAL command, although this setting is not recommended for normal product processing.
DBRC function used by Fast Path/EP Series products

When DBRC is active during the execution of a Fast Path Indexer/EP utility function, it is used for the following functions:

- **access control**
  
  Fast Path Indexer/EP requests access authorization for each area prior to its use. When the product is finished processing the area, it requests deauthorization of the area. The authorization level used for the area depends on the requirements of the product function that is executing.

- **dynamic allocation of area data sets**
  
  The Fast Path/EP Series products use the registered name of the area data sets for dynamic allocation. If multiple area data sets (MADS) are specified for an area, then Fast Path Indexer/EP uses the first data set marked as available in the ADS list.

- **recovery control**
  
  Fast Path Indexer/EP products record events, as necessary, that cause a change in the recovery status of the area.

- **image copy notification**
  
  Fast Path Indexer/EP records the successful creation of image copy data sets.

**RECON data sets**

If the Fast Path Indexer/EP utility function executing in offline mode uses DBRC during processing, the function requires access to the RECON data sets.

If the RECON data sets are not supplied in the execution JCL, they are dynamically allocated by DBRC, using the appropriate DFSMDA members loaded from the STEPLIB. This is the recommended method.

The RECON data sets can be specified in the execution JCL as shown in Figure 46.
Caching in offline mode

Fast Path Indexer/EP provides a caching mechanism for the index build and verify processes that can improve performance in some instances. The caching mechanism is used to process significant portions of the area within virtual storage (using a data space) rather than using physical read and write I/O operations to DASD.

Multiple area data sets

Multiple area data sets (MADS) are not supported in offline mode. Fast Path/EP Series products process only one area data set registered in the ADS list for an area.

For functions that read the database but do not update it, the ADS list registered for each area is searched (in collating sequence by DD name) for the first ADS that is marked as available for use and that has no error queue elements (EQEs). The selected ADS is read and processed; all other area data sets are ignored.

For functions that read and update the database, the ADS list registered for each area is searched (in collating sequence by DD name) for the first ADS that is marked as available for use and that has no EQEs. The selected ADS is read and updated; all other area data sets are marked as unavailable.

For functions that create the database, the first ADS (in collating sequence by DD name) is selected and created. All other area data sets are marked as unavailable.

When functions that update or create the database have completed successfully, only one ADS will be marked as available. Use the IBM Online MADS Create utility to resynchronize the other (unavailable) area data sets.

Caching in offline mode

Fast Path Indexer/EP provides a caching mechanism for the index build and verify processes that can improve performance in some instances. The caching mechanism is used to process significant portions of the area within virtual storage (using a data space) rather than using physical read and write I/O operations to DASD.
The caching function is most beneficial when the product processes control intervals in a random (non-sequential) manner. The cache is loaded by using a high-speed sequential read process, and unloaded (if necessary) by using a sequential write process. Because random processing is performed within the data space, the time required to perform random I/O operations to DASD is significantly reduced. This advantage is offset, however, by the resources that are required to maintain the data space within virtual storage.

It is neither necessary nor beneficial to load the entire area into a cache. The base UOW control intervals (RAA and DOVF blocks), for example, are always processed sequentially. Caching does not provide any benefit for sequential processing. However, if functions randomly access IOVF control intervals, a potential benefit might result from caching IOVF blocks; for more information, see the performance considerations for each product function.

**Caching for input areas**

When caching is requested for an input area (ICACHE), portions of the area data set are read sequentially into a data space. As the product processes the cached control intervals, the control intervals are read from or written to the data space rather than DASD. When product processing completes, any control intervals that have been modified within the data space are written sequentially to DASD.

**System resource limits**

When an area or portions of an area are cached, a data space is created. A data space cannot exceed 2 GB, and must be backed by expanded storage and/or paging space. Fast Path Indexer/EP uses a compression technique to minimize the size of the data space, but the total amount of data space storage that is available is frequently a limited resource.

Fast Path Indexer/EP queries the operating system regarding the amount of data space storage available. The product does not attempt to create a data space that would exceed the capacity of the extended storage and page space that are available to the operating system.

**Online (BMP) processing**

The primary commands (VERIFY and RESYNC) that are associated with the Fast Path Indexer/EP verify and resynchronize functions can be executed in an IMS batch message processing (BMP) region. BMP mode is characterized by processing DEDB areas that are online to one (or more) IMS control regions. DEDB areas that are processed by the Fast Path Indexer/EP product functions in BMP mode are available to applications executing simultaneously.
Execution JCL for BMP processing

Figure 47 shows the basic JCL that is required for execution of the Fast Path Indexer/EP product verify and resynchronize functions in BMP mode. This JCL assumes dynamic allocation of the OLCSTAT, MODSTAT, IMSACBA, IMSACBB, PFXLIBA, and PFXLIBB data sets, which are required.

**Figure 47  BMP mode execution JCL**

```plaintext
//PFP EXEC PGM=DFSRRC00,REGION=0M,
//       PARM=(BMP,PFPMAIN,psbname)
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
//       DD DISP=SHR,DSN=IMSVS.RESLIB
//PFPSYSIN DD *
--- control statements ---
/*

Other parameter values for the PARM keyword might be required for BMP regions at your site. The standard catalogued procedure that is supplied with IMS (IMSBATCH) can be used for Fast Path Indexer/EP.

---

**NOTE**

The PFXLIB, PFXLIBA, and PFXLIBB DD statements are only needed for processing PFX indexes.

---

Region size

To ensure optimum performance, BMC recommends that you request the maximum available storage for execution of the Fast Path Indexer/EP utility functions. Maximum available storage is normally requested by specifying REGION=0M on the EXEC statement for the JOB step. Some installations might have different requirements. Check with your MVS systems programmer to determine how to request the maximum REGION size.

STEPLIB requirements

BMC recommends that the PFP installation library appear first within the STEPLIB concatenation. If any other library precedes the product installation library, you must ensure that this library does not contain a member name that duplicates a member within the PFP product library. It is not necessary that all data sets listed within the STEPLIB concatenation be APF authorized.
Identifying the functions to be performed

The PFPSYSIN control statements identify the functions to be performed. The DBD keyword is required for BMP functions and identifies the database (DBD name) to be processed by the command. The DBD name must be contained within the PSB specified in the PARM keyword on the EXEC statement for the JOB step. The processing options (PROCOPT) for the database must be consistent with the function being requested. Multiple BMP function commands can be specified within the PFPSYSIN control statements.

NOTE
No parallel processing of commands is performed in BMP execution mode. Each command is executed serially in the order specified.

Creating the PSB for BMP processing

The following requirements apply for creating the program specification block (PSB) for BMP processing:

- You must create a separate PCB statement for the primary DBD and each index DBD. If an index is partitioned, each partition DBD must have a separate PCB statement.

- Each index PCB statement and partitioned index PCB statement must be specified in the order listed in the primary DBD.

- The PCB statement for the primary DEDB must have SENSEG statements for the hierarchical path for the source segments.

- The PCB statement for each secondary index database must have SENSEG statements for the segment specified in the DBD.

- For the VERIFY command, the PROCOPT should be specified as read (G). For the RESYNC command, the PROCOPT must have read and write (A) capability.

- The LANG parameter in the PSBGEN statement must be set to COBOL or ASM.

Identifying the DEDB area to be processed

In BMP mode, the IAREA keyword must not be specified.
Access to online data sets

The Fast Path Indexer/EP product functions that are executing in the BMP mode access the data sets by using IMS control region services. All facilities of the online IMS control region are supported, including Multiple Area Data Sets (MADS) and the Virtual Storage Option (VSO). No areaname DD statement is required, and any areaname DD statement that is supplied is ignored.

The functions access the RECON data sets by using DBRC region services. The RECON1, RECON2, and RECON3 DD statements are not required. If any of these statements is supplied, it is ignored.

IMS commands

IMS/VS commands are available to monitor and control an online DEDB during Fast Path Indexer/EP BMP function processing.

- /DISPLAY ACTIVE lets you to display the status of all active IMS regions. This display includes the BMP region that is used to execute the Fast Path/EP Series online functions.

- /STOP REGION regid lets you to terminate the Fast Path region at the next utility checkpoint. regid is the region ID number that is assigned to the Fast Path Region. You can determine the regid by displaying it with the /DISPLAY ACTIVE command.

**WARNING**

/STOP REGION ABDUMP or CANCEL can cause a U113 abend in the IMS control region.

Online (IFP) processing

The primary command (XSCAN) that is associated with the Fast Path Indexer/EP product XSCAN function can be executed in an interactive Fast Path (IFP) region. IFP mode is characterized by processing DEDB areas that are online to one (or more) IMS control regions. DEDB areas that are processed by the Fast Path Indexer/EP product functions in IFP mode are available to applications executing simultaneously. However, only one area at a time can be processed when using IFP mode.
Execution JCL for IFP processing

When processing in an IFP region, IMSACB and PFXLIB are required. The ACB member for the primary database comes from the online IMS system. However, the ACB members for the indexes must be loaded within the IFP region.

If the DD statements for OLCSTAT, MODSTAT, IMSACBA, IMSACBB, PFXLIBA, and PFXLIBB are not coded in the JCL, the JCL assumes dynamic allocation of the required OLCSTAT, MODSTAT, IMSACBA, IMSACBB, PFXLIBA, and PFXLIBB data sets.

Figure 48 shows the basic JCL that is required for execution of a Fast Path Indexer/EP XSCAN function in IFP mode. Because the DD statements are not coded in the JCL, the JCL assumes dynamic allocation of the OLCSTAT, MODSTAT, IMSACBA, IMSACBB, PFXLIBA, and PFXLIBB data sets.

Figure 48 IFP mode execution JCL

```
//PFP  EXEC  PGM=DFSRRC00,REGION=0M,
//       PARM=(IFP,dbname,DFU#PUO)
//STEPLIB  DD DISP=SHR,DSN=BMC.PFP.LOAD
//         DD DISP=SHR,DSN=IMSVS.RESLIB
//         DD DISP=SHR,DSN=BMC.DPK.LOAD
//$$DPICDS DD DISP=SHR,DSN=PFP.DPK.DPICDS
//$$DPITBL DD DISP=SHR,DSN=PFP.DPK.DPITBL
//PFPSYSIN DD *
--- control statements ---
/*
```

Other parameter values for the PARM keyword might be required for IFP regions at your site. The standard catalogued procedure that is supplied with IMS (FPUTIL) can be used for Fast Path Indexer/EP.

**NOTE**

The PFXLIB, PFXLIBA, and PFXLIBB DD statements are only needed for processing PFX indexes.
Region size

To ensure optimum performance, BMC recommends that you request the maximum available storage for execution of the Fast Path Indexer/EP utility functions. Maximum available storage is normally requested by specifying REGION=0M on the EXEC statement for the JOB step. Some installations might have different requirements. Check with your z/OS systems programmer to determine how to request the maximum REGION size.

STEPLIB requirements

BMC recommends that the PFP installation library appear first within the STEPLIB concatenation. If any other library precedes the product installation library, you must ensure that this library does not contain a member name that duplicates a member within the PFP product library. It is not necessary that all data sets listed within the STEPLIB concatenation be APF authorized.

Identifying the functions to be performed

The PFPSYSIN control statements identify the functions to be performed. The DBD keyword is not required for IFP functions since the database (DBD name) is specified in the PARM keyword on the EXEC statement for the JOB step.

Identifying the DEDB area to be processed

In IFP mode, if you want to process data only for specific areas within the DEDB, use the IAREA keyword to specify the name of the area or areas to be used as input. You can specify one or more area names. If you omit the IAREA keyword, all areas that are defined in the DEDB are used as input to the process.

Areas might be specified on the IAREA keyword by using any combination of area names, area numbers, or area ranges. Table 18 lists the parameters that are available for the IAREA keyword.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>specify all areas of the DEDB (default)</td>
</tr>
</tbody>
</table>
To specify all areas in a DEDB as input to the XSCAN process, use a command set like the example shown in Figure 49.

**Figure 49  Sample control statement for specifying all areas as input to XSCAN**

```
XSCAN DBD=dbdname,IAREA=ALL
```

To specify only selected areas as input to the XSCAN process, use a command set like the example shown in Figure 50.

**Figure 50  Sample control statement for specifying selected areas as input to XSCAN**

```
XSCAN DBD=dbdname,IAREA=(AREANAM1,AREANAM3)
```

This control statement uses a combination of area names and area numbers to request that the specified areas are to be input to the index XSCAN process. The UOWs in areas named AREANAM1 and AREANAM3 will be input.

### Access to online data sets

The minimum JCL for executing the Fast Path/EP Series product functions in online mode is simple: many of the necessary data sets can be allocated dynamically. In contrast, adding DD statements to the JCL uses the specified data sets instead of dynamic allocation. BMC recommends omitting those DD statements so that dynamic allocation is used.
IMS ACB library

Executing Fast Path Indexer/EP utility functions in online mode requires access to the IMS ACB library containing the database definitions (DMB) for the DEDB areas to be processed. If you omit the IMSACB DD statement from the JCL, Fast Path Indexer/EP attempts to allocate the DD. The following IMSACB DD statements identify the active ACB library:

- IMSACB
- IMSACBA
- IMSACBB
- OLCSTAT
- MODSTAT
- MODSTAT2
- PFXLIB
- PFXLIBA
- PFXLIBB

**NOTE**
The PFXLIB, PFXLIBA, and PFXLIBB DD statements are necessary for online functions that process PFX indexes.

For details about using DD statements to identify the IMS ACB library, see the *IMS/ESA System Administration Guide*.

**NOTE**
The IMS/ESA release level of the ACB library must be the same as that of the RESLIB that is included in the STEPLIB DD statement concatenation.

Online database data sets

All facilities of the online IMS control region are supported, including Multiple Area Data Sets (MADS) and the Virtual Storage Option (VSO). No areaname DD statement is needed, and any areaname DD statement that is supplied is ignored.

RECON data sets

The functions access the RECON data sets by using DBRC region services. The RECON1, RECON2, and RECON3 DD statements are not needed, and any of these statements that are supplied are ignored.
Index scan utility

This chapter provides information about the capabilities and use of the optional Fast Path Indexer/EP scan (XSCAN) function. This function is executed by a command-driven utility that lets you scan a Fast Path primary DEDB to locate index source segments and then produce an output data set containing index information.

**NOTE**

All topics in this chapter apply to both PFX indexes and IBM native indexes, except where noted.

This chapter discusses the following topics:

- Introduction ................................................................. 142
- XSCAN function inputs and outputs ................................ 142
- DBRC considerations ...................................................... 143
  - Offline mode ............................................................. 143
  - IFP mode ................................................................. 143
- Index XSCAN command language ...................................... 143
- Selecting the database and areas ...................................... 144
- Allocating the source area data set .................................. 144
  - Offline mode ............................................................. 145
  - IFP mode ................................................................. 145
- Specifying secondary index databases to be scanned ............. 145
  - Using the INDEX keyword or the &INDEX parameter .......... 146
  - Using the DDNAME or DSNAME keyword ......................... 146
- Allocating the XSCAN data set ........................................ 147
- Performance considerations for the XSCAN function ............... 148
  - Offline mode ............................................................. 148
  - IFP mode ................................................................. 148
- Sorting index information for the XSCAN function ................. 148
  - Using the SORT keyword ............................................. 149
  - Using the SORT_OPTION keyword ................................. 149
- Sample XSCAN command scenarios .................................... 149
  - Scenario 1: Offline scan of all indexes defined to a DEDB ...... 150
  - Scenario 2: Offline scan for specified indexes defined to a DEDB 150
  - Scenario 3: Online (IFP) scan of all indexes defined to a DEDB 151
Introduction

By using the command-driven XSCAN utility, you can scan a single area or multiple areas in a primary DEDB in offline or online mode. Fast Path Indexer/EP executes an online scan in an interactive Fast Path (IFP) region.

Fast Path Indexer/EP provides the XSCAN command, the IX subcommand, and the associated keywords. The XSCAN command (when used with the IX subcommand and associated keywords) lets you scan a single area or multiple areas in a primary DEDB to locate index source segments for a defined index and then produce an output data set containing index information. This output data set can be used as input into the build, verify, and resynchronize (can only process PFX indexes) utilities.

XSCAN function inputs and outputs

The XSCAN command can accept input from an entire DEDB or from specific areas within a DEDB. The output from the XSCAN command is a sequential data set that contains index information which can be used as input into the build, verify, and resynchronize utilities. Inputs and outputs are shown in Figure 51.

Figure 51  XSCAN function inputs and outputs
Fast Path Indexer/EP generates a report containing a summary of statistics when the scan activities are complete.

**DBRC considerations**

This section discusses DBRC considerations for the index XSCAN function.

**Offline mode**

If your IMS system has been defined to force the use of DBRC (DFSIEDF0 TYPE=PARM, DBRC=FORCE), this option might need to be overridden in offline mode.

**IFP mode**

When the index XSCAN process is executed in IFP mode, all databases must be online and available for input processing.

**Index XSCAN command language**

Table 19 lists the keywords and subcommands that are available for the XSCAN command.

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the primary DEDB and areas (for input to the index XSCAN process)</td>
<td>XSCAN</td>
<td>DBD, IAREA</td>
</tr>
<tr>
<td>allocating the source area data set (for input to index XSCAN process)</td>
<td>XSCAN</td>
<td>INPUT_DSN_MASK</td>
</tr>
<tr>
<td>specifying the indexes for which you want XSCAN to create output</td>
<td>IX</td>
<td>INDEX</td>
</tr>
</tbody>
</table>
Selecting the database and areas

The DBD keyword identifies the name of the primary DEDB (DBD name) for which you want to scan a primary DEDB for all source segments for an index or indexes. The DBD will be loaded from the corresponding member in the IMSACB library.

In offline mode and IFP mode, if you want to scan data only for specific areas within the DEDB, use the IAREA keyword to specify the name of the area or areas to be used as input. You can specify one or more area names. If you omit the IAREA keyword, all areas that are defined in the DEDB are used as input to the process. For more information, see “Identifying the DEDB areas to be processed” on page 128.

Allocating the source area data set

This section discusses considerations for allocating the source area data set for the XSCAN function in offline mode and online mode.

Table 19  XSCAN command keywords and subcommands (part 2 of 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocating the XSCAN data set</td>
<td>IX</td>
<td>DNAME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSNAME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Related keywords:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AVGREC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATACLAS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXPDT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LIKE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MGMTCLAS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETPD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STORCLAS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VOLCNT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VOLSER</td>
</tr>
<tr>
<td>enhancing performance of the XSCAN process</td>
<td>XSCAN</td>
<td>ICACHE</td>
</tr>
<tr>
<td>sorting index information</td>
<td>IX</td>
<td>SORT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SORT_OPTION</td>
</tr>
</tbody>
</table>

For syntax of the commands, subcommands, and keywords shown in Table 19, see the Fast Path/EP Series Reference Manual.
Offline mode

For offline mode processing, the area data sets can be supplied in the JCL or can be accessed by using dynamic allocation. Each areaname DD statement identifies an area data set to be read. If the areaname DD statement is omitted from the JCL, Fast Path Indexer/EP attempts to dynamically allocate it.

The IMSACB DD statement identifies the ACB library containing the database definition that describes the area referenced by the areaname DD statement.

If you are using dynamic allocation, do not include the areaname DD statement. Fast Path Indexer/EP attempts to obtain the data set name for allocation in the following order:

1. If the INPUT_DSN_MASK keyword is specified, it is used to generate the data set name.
2. If DBRC is active and the area is registered, the registered area data set name is obtained from DBRC.
3. The STEPLIB is searched for the DFSMDA member that contains the data set name for this area.

If DBRC is active, and the area is registered with DBRC, then the allocated data set name must match the registered data set name regardless of how it is allocated.

IFP mode

For IFP mode processing, the area data set and secondary index databases are accessed using IMS services. No JCL is needed for the area data set, and it is ignored if present.

Specifying secondary index databases to be scanned

Use the IX subcommand with the XSCAN command to create data for a secondary index database. At least one IX subcommand is required. By default, data for all secondary index databases that are registered to the primary DEDB is created. The example in Figure 52 shows how to use a single IX subcommand to create data for all secondary index databases for the primary DEDB.
Using the INDEX keyword or the &INDEX parameter

You can use the INDEX keyword to create data for a single secondary index database or for multiple secondary index databases. Use one IX subcommand with the INDEX keyword for each secondary index database that needs data created. The example in Figure 53 shows how to create data for a single secondary index database to a primary DEDB.

You can also use the &INDEX substitution parameter to create unique data set names when processing multiple secondary index databases. The substituted value for &INDEX is the name of the index being processed by Fast Path Indexer/EP when the mask of &INDEX is used with the DSNAME keyword on the IX subcommand.

Since the INDEX keyword is not specified, all indexes that are defined for the database are processed. Four output data sets are generated, one for each index (for example, PFXIDX1, PFXIDX2, PFXIDX3, and PFXIDX4), each having a unique data set name. The example in Figure 54 shows how to create data for multiple secondary index databases to source a database by using the &INDEX parameter.

Using the DDNAME or DSNAME keyword

Either the DDNAME or the DSNAME keyword must also be specified on the IX subcommand to generate the output XSCAN data set. The XSCAN data set contains index information created by the XSCAN utility. A separate XSCAN data set must be created for each index that is specified. For more information about using the DDNAME keyword or the DSNAME keyword to generate the XSCAN data set, see “Allocating the XSCAN data set.”
Allocating the XSCAN data set

The XSCAN data set that is created by the XSCAN utility can be used for input into the build, verify, and resynchronize functions. A separate XSCAN data set must be created for each index that is specified.

When processing the XSCAN function in offline mode or IFP mode, the XSCAN data set can be supplied in the JCL, or it can be accessed by using dynamic allocation. If you want to use JCL to supply the XSCAN data set, specify the DDNAME keyword with the IX subcommand. If you want the XSCAN data set to be dynamically allocated, specify the DSNAME keyword with the IX subcommand.

For dynamic allocation, you can also use the DISP keyword to control the allocation and disposition of the data set. The DISP=USE status parameter indicates conditional allocation. If the data set does not exist, it is created (as if DISP=NEW had been specified). If the data set already exists, it is reallocated (as if DISP=OLD had been specified). If CATLG is specified, the normal and conditional parameters are changed from CATLG to KEEP. In addition to the DISP keyword, the following keywords can also be used to control the allocation of a data set:

- AVGREC
- SPACE
- DATACLAS
- STORCLAS
- EXPDT
- UNIT
- LIKE
- VOLCNT
- MGMTCLAS
- VOLSER
- RETPD

For more information about these keywords, see the *Fast Path/EP Series Reference Manual*. 
Performance considerations for the XSCAN function

The following performance considerations apply to the XSCAN function.

Offline mode

When running an XSCAN in offline mode, you can use input caching to improve performance. You can specify input caching of the primary DEDB or areas by using the ICACHE keyword. Input caching can enhance the processing of the XSCAN function. Input caching of IOVF CIs can significantly enhance performance if the following conditions are present:

- The DEDB has a significant number of IOVF CIs that are likely to be processed by the function.
- Storage resources are available to support storage of all IOVF CIs.

IFP mode

When executing an XSCAN in IFP mode, Fast Path Indexer/EP uses IMS services to read the primary DEDB. As with any application IFP, the execution time is governed by the speed of IMS calls and logging services. Scanning a primary DEDB in IFP mode require much more time to run than scanning a primary DEDB in offline mode.

Sorting index information for the XSCAN function

This section discusses how to use the SORT keyword and the SORT_OPTION keyword to sort index information when using an XSCAN data set created by the XSCAN utility.
Using the SORT keyword

When used with the IX subcommand, the SORT keyword can be used to specify whether the scanned index records are to be sorted into index key sequence. The default, SORT=YES, automatically sorts the scanned index records into index key sequence. If SORT=NO is specified, the index records are not sorted into index key sequence.

The example in Figure 55 shows how to sort scanned index records into index key sequence when either the DDNAME or DSNAME is also specified on the IX subcommand. For more information, see “Allocating the XSCAN data set” on page 147 and the Fast Path/EP Series Reference Manual.

Using the SORT_OPTION keyword

When used with the IX subcommand, the SORT_OPTION keyword can be used to specify a list of sort utility control options. The SORT_OPTION keyword can only be used with SORT=YES (default); it cannot be used if SORT=NO is specified. The YES parameter provides a list of sort utility options. For more information about sort utility control options, see the reference manual for the sort product that is used at your site.

The example in Figure 55 shows how to sort index information records into index key sequence when either the DDNAME or DSNAME is also specified on the IX subcommand, and how to specify a list of sort utility control options. For more information, see “Allocating the XSCAN data set” on page 147 and the Fast Path/EP Series Reference Manual.

Figure 55  Sorting index information by using SORT and SORT_OPTION keywords

<table>
<thead>
<tr>
<th>XSCAN</th>
<th>DBD=dbdname, IAREA=ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>IX</td>
<td>INDEX=index1,</td>
</tr>
<tr>
<td></td>
<td>[DDNAME=ddname,]</td>
</tr>
<tr>
<td></td>
<td>[SORT=YES</td>
</tr>
</tbody>
</table>

Sample XSCAN command scenarios

The scenarios in this section illustrate how to use the XSCAN command in offline mode and online (IFP) mode.
### Scenario 1: Offline scan of all indexes defined to a DEDB

In the example shown in Figure 56, the primary DEDB DIVSSD is input into the offline XSCAN process. Four indexes are defined for the primary DEDB: two are in SHISAM format and two are in HISAM format. The IAREA keyword is omitted from the XSCAN command. As a result, the default value of IAREA=ALL will be used and all areas will be scanned. Input area data sets use DFSMDA members for dynamic allocation.

Because the INDEX keyword is not specified on the IX subcommand, all four indexes that are defined for DIVSSD will be scanned, and four separate output data sets will be created. The DSNAME keyword identifies the sequential output data set where the corresponding index information records are written. In the example in Figure 56, the &DBD and &INDEX variables represent the names of the DBD and index.

**Figure 56 Offline scan of all indexes defined to a DEDB**

```
//XSCAN EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// DD DISP=SHR,DSN=IMSVS.RESLIB
//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
//PFXLIB DD DISP=SHR,DSN=BMC.PFXLIB
//PFPRINTF DD SYSOUT=*  
//PFPRPTS DD SYSOUT=*  
//PFPSYSIN DD *
GLOBAL DBRC=YES
XSCAN DBD=DIVSSD
   IX DSNAME='BMC.&DBD.&INDEX.XSCAN',
      SPACE=(CYL,10,1),UNIT=SYSDA,DISP=(USE,CATLG)
```

### Scenario 2: Offline scan for specified indexes defined to a DEDB

In the example in Figure 57, the primary DEDB EIVSSD is input into the offline XSCAN process. Six indexes are defined for the primary DEDB, but only index EIVSI3 and EIVSI5 will be included in the process. An IX subcommand is needed for each index. The indexes will be processed simultaneously.

---

**NOTE**
The PFXLIB, PFXLIBA, and PFXLIBB DD statements in the following examples are only needed for processing PFX indexes.
The INDEX keyword on the IX subcommand specifies the name of the index. The DSNAME keyword identifies the sequential output data set where the corresponding index information records are written.

**Scenario 3: Online (IFP) scan of all indexes defined to a DEDB**

In the example shown in Figure 58, the primary DEDB DIVSSD is input into the offline XSCAN process. Four indexes are defined for the primary DEDB: two are in SHISAM format and two are in HISAM format. The IAREA keyword is omitted from the XSCAN command. As a result, the default value of IAREA=ALL will be used and all areas will be scanned.

Because the INDEX keyword is not specified on the IX subcommand, all four indexes defined for DIVSSD will be scanned, and four separate output data set will be created. The DSNAME keyword identifies the sequential output data set where the corresponding index information records are written. The &DBD and &INDEX variables represent the names of the DBD and index.

When processing in an IFP region, the ACB member for the primary DEDB database comes from the online IMS system. However, the ACB members for the SHISAM and HISAM indexes must be loaded within the IFP region.

Unless the DD statements are coded in the JCL (like the example in Figure 58), the JCL assumes dynamic allocation of the required OLCSTAT or MODSTAT, IMSACBA, IMSACBB, PFXLIBA, and PFXLIBB data sets.
Scenario 3: Online (IFP) scan of all indexes defined to a DEDB

For all of the online scenarios, specifying the DBD name in the PPFSYSIN control card statement is not required because the DBD name is taken from the PARM statement on the EXEC statement.

Figure 58  Online scan for specified indexes defined to a DEDB

```plaintext
//XSCAN EXEC PGM=DFSRRC00,REGION=0M,
   PARM=(IFP,DIVSSD,DBF#FPU0)
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// DD DISP=SHR,DSN=IMSVS.RESLIB
// DD DISP=SHR,DSN=BMC.DPK.LOAD
//$$DPICDS DD DISP=SHR,DSN=PFP.DPK.DPICDS
//$$DPITBL DD DISP=SHR,DSN=PFP.DPK.DPITBL
//MODSTAT DD DISP=SHR,DSN=IMSVS.MODSTAT
// IMSACBA DD DISP=SHR,DSN=IMSVS.ACBLIBA
// IMSACBB DD DISP=SHR,DSN=IMSVS.ACBLIBB
//PFXLIBA DD DISP=SHR,DSN=BMC.PFXLIBA
//PFXLIBB DD DISP=SHR,DSN=BMC.PFXLIBB
//PFPRINTER DD SYSOUT=* 
//PFPRPTS DD SYSOUT=* 
//PPFSYSIN DD *
   GLOBAL DBRC=YES
XSCAN
   IX DSNAME='BMC.&DBD.&INDEX.XSCAN',
      SPACE=(CYL,10,1),UNIT=SYSDA,DISP=(NEW,CATLG)
```
This chapter provides information about the capabilities and use of the Fast Path Indexer/EP index build function. This function is executed by a command-driven utility that lets you create an index to a Fast Path primary DEDB.

**NOTE**

All topics in this chapter apply to both PFX indexes and IBM native indexes.

This chapter discusses the following topics:

- Introduction .......................................................... 154
- Preparing for the index build ........................................ 154
- Build function inputs and outputs .................................. 155
- DBRC considerations .................................................. 155
- BUILD command keywords and subcommands ....................... 156
- Selecting the database and areas ..................................... 157
- Input to the build process ........................................... 158
  - Allocating the DEDB area data set ............................. 158
  - Allocating the XSCAN data set ................................. 158
- Allocating the secondary index databases .......................... 159
- Specifying indexes to be built ..................................... 159
  - Using the INDEX keyword or the &INDEX parameter ............ 160
- Index partitioning ..................................................... 160
- Performance considerations ........................................ 162
  - Input caching ...................................................... 162
  - Input thread processing ......................................... 162
  - Index thread processing ......................................... 163
- Customizing sort processing ......................................... 163
- Sorting index information for the build function .................... 164
  - Using the SORT keyword ........................................ 164
  - Using the SORT_OPTION keyword ............................. 164
- Rebuilding indexes during database change ........................ 165
- Rebuilding indexes during database reload ........................ 166
- Sample BUILD command scenarios .................................. 166
Introduction

By using the Fast Path Indexer/EP index build function, you can build a single index or multiple indexes to a primary DEDB in offline mode only. You can build indexes by reading the primary DEDB, or by using an XSCAN data set that the XSCAN command creates. The two approaches for building indexes are mutually exclusive.

Fast Path Indexer/EP provides the BUILD command, the IX subcommand, and associated keywords. You can use them in various combinations to build indexes to a primary DEDB. You can also use the BUILD command in the same manner to rebuild indexes following maintenance to the primary DEDB.

For IMS Version 12 and later, you can use the BUILD command to build PFX indexes or IBM native indexes.

Preparing for the index build

Before building an index, you must modify the primary DEDB to include Fast Path Indexer/EP PFX macros or IBM native index macros that define the index. A Fast Path Indexer/EP utility, PFXAGEN, provides a front-end process to the IMS ACBGEN. You must run PFXAGEN to create the required control blocks for PFX indexes. You can also use PFXAGEN or ACBGEN to create control blocks for IBM native indexes.

This chapter provides an overview of how to use the BUILD command and its associated subcommands and keywords to build (or rebuild) an index to a DEDB. When all steps of the index registration process have been completed, and after the index has been built, it can then be used in the IMS environment.

When DBD changes that affect the index registration have been made against the primary DEDB, you can execute the BUILD command again to rebuild the index. You can execute BUILD in offline mode only.
Build function inputs and outputs

The BUILD command can accept input from an entire DEDB, from specific areas within a DEDB, or it can accept input from an XSCAN data set that is created by the XSCAN command. The output from the BUILD command is an index or indexes that are associated with the primary DEDB. Inputs and outputs are shown in Figure 59.

Figure 59  Build function inputs and outputs

When an index build has been completed, Fast Path Indexer/EP generates a report containing a summary of statistics. For an example of the Build Summary Report, see the PFPBLD report in the REPORTS data set.

DBRC considerations

When the index build function executes, the index is built directly from a primary DEDB, and DBRC is active during execution. For this execution, Fast Path Indexer/EP obtains authorization for each registered database. The product obtains authorization for “read with integrity” (RD) from DBRC for each input area. It also obtains exclusive access (EX) for each index to be built. These conditions are listed in Table 20.
Multiple area data sets (MADS) are not supported for the source input areas. When executing an index build, Fast Path Indexer/EP searches the ADS (area data set) list registered for each area (in collating sequence by DD name). The product selects the first ADS that is marked as available for use and that has no error queue elements (EQEs). If an ADS is found that meets both of these criteria, it is the only ADS that will be used by Fast Path Indexer/EP. All other area data sets are ignored. If no area data set is marked as available, or if all available area data sets contain one or more EQEs, the build function cannot be performed.

**BUILD command keywords and subcommands**

The keywords and subcommands that are available for the BUILD command are shown in Table 21.

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the primary DEDB and areas (for input to the index build process)</td>
<td>BUILD</td>
<td>DBD</td>
</tr>
<tr>
<td></td>
<td>IAREA</td>
<td></td>
</tr>
<tr>
<td>allocating the source area data set</td>
<td>BUILD</td>
<td>INPUT_DSN_MASK</td>
</tr>
<tr>
<td>specifying indexes to be built</td>
<td>IX</td>
<td>INDEX</td>
</tr>
<tr>
<td>allocating the index data sets</td>
<td>IX</td>
<td>OUTPUT_DSN_MASK</td>
</tr>
<tr>
<td>processing the XSCAN data set</td>
<td>IX</td>
<td>DDNAME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSNNAME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Related keywords:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISP, UNIT, VOLSER</td>
</tr>
<tr>
<td>enhancing performance</td>
<td>BUILD</td>
<td>INPUT_THREADS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ICACHE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INDEX_THREADS</td>
</tr>
<tr>
<td>customizing sort processing</td>
<td>BUILD</td>
<td>SORT_OPTION</td>
</tr>
<tr>
<td>sorting index information</td>
<td>IX</td>
<td>SORT_OPTION</td>
</tr>
</tbody>
</table>
Selecting the database and areas

The DBD keyword identifies the name of the primary DEDB (DBD name) for which you want to create an associated index or indexes. The DBD will be loaded from the corresponding member in the IMSACB library.

If you want to build indexes only for specific areas within the DEDB, use the IAREA keyword to specify the name of the area or areas to be used as input. You can specify one or more area names. If you omit the IAREA keyword, all areas defined in the DEDB are used as input to the process.

NOTE
If you use a sequential input data set containing index information (created by the XSCAN utility) for input into the build process, you cannot use the IAREA keyword to specify the name of the area or areas to be used as input into the build process.

Areas might be specified on the IAREA keyword by using any combination of area names, numbers, or ranges. For a list of the parameters that are available for the IAREA keyword, see Table 18 on page 138.

To specify all areas in a DEDB as input to the XSCAN process, use a command set like the example shown in Figure 60.

**Figure 60  Sample control statement for specifying all areas as input to the index build**

```
BUILD DBD=dbdbname,IAREA=ALL
```

To specify only selected areas as input to the index build process, use a command set like the example shown in Figure 61.

**Figure 61  Sample control statement for specifying selected areas as input to the index build**

```
BUILD DBD=dbdbname,IAREA=(AREANAM1,AREANAM3,RANGE=(5,8))
```

This control statement uses a combination of area names and area numbers to request that the specified areas are to be input to the index XSCAN process. The UOWs in areas named AREANAM1 and AREANAM3 will be input. In addition, all consecutive areas from area number 5 through area number 8 (area5, area6, area7, and area8) will be input.
Input to the build process

This section discusses considerations for allocating the input data set for the index build function. The following input options for building the index are available:

- DEDB area data set
- XSCAN data set

Allocating the DEDB area data set

For information about allocating the DEDB area data set, see “Allocating the source area data set” on page 144.

Allocating the XSCAN data set

You can allocate the XSCAN data set dynamically or specify it in the JCL. If you want to allocate it dynamically, you must specify the DSNAME keyword on the IX subcommand. If you specify the XSCAN data set in the JCL, you must use the DDNAME keyword on the IX subcommand:

- If you specify the DSNAME keyword on the IX subcommand, the product dynamically allocates the XSCAN data set. You must specify the DISP keyword, as shown in the following example:

```
BUILD DBD=dbname,AREA=ALL
IX DSNAME='PFP.&DBD.&INDEX.XSCAN',DISP=SHR
```

- If you specify the DDNAME keyword on the IX subcommand, you must include the XSCAN data set on the DD statement that is specified by the DDNAME keyword in the JCL. An example follows:

```
//XSCAN1 DD DSN=datasetname
//PFPSYSIN DD *
BUILD DBD=dbname,AREA=ALL
   IX DSNAME=XSCAN1
```

- If you do not specify the DSNAME or DDNAME keyword on the IX subcommand, the product uses the default scan of the primary DEDB to create the required index information records.
Allocating the secondary index databases

You can specify the secondary index database in the JCL, or let it be allocated dynamically.

The indexname DD statement identifies the secondary index database to be created. If you omit the indexname DD statement from the JCL, Fast Path/EP attempts to dynamically allocate it. The IMSACB DD statement identifies the ACB library containing the database definition that describes the index referenced by the indexname DD statement.

If DBRC is active, and the secondary index database is registered with DBRC, the data set that is named on the indexname DD statement must match the registered data set name.

If you are using dynamic allocation, do not include the indexname DD statement. Fast Path Indexer/EP attempts to obtain the data set name for allocation in the following order:

1. If the OUTPUT_DSN_MASK keyword is specified on the IX subcommand, Fast Path Indexer/EP uses that keyword to generate the data set name for the index.
2. If DBRC is active and the secondary index database is registered, the registered data set name is obtained from DBRC.
3. The STEPLIB is searched for the DFSMDA member that contains the data set name for this index.

Specifying indexes to be built

Use the IX subcommand with the BUILD command to specify which indexes are to be built. At least one IX subcommand is required. By default, all indexes that are registered to the primary DEDB are built. The example in Figure 62 shows how to use a single IX subcommand to build all indexes for the primary DEDB.

Figure 62  Building all indexes to a primary DEDB

<table>
<thead>
<tr>
<th>BUILD</th>
<th>DBD=dbdname,IAREA=ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IX</td>
</tr>
</tbody>
</table>
Using the INDEX keyword or the &INDEX parameter

You can use the INDEX keyword to specify a single index or multiple indexes to be built. Use one IX subcommand with the INDEX keyword for each index to be built. The example in Figure 63 shows how to build a single index to a DEDB.

**Figure 63  Building a single index to a DEDB**

```
BUILD DBD=dbdname, IAREA=ALL
  IX INDEX=index1
```

The example in Figure 64 shows how to specify multiple IX subcommands.

**Figure 64  Specifying multiple IX subcommands**

```
BUILD DBD=dbdname, IAREA=ALL
  IX INDEX=index1
  IX INDEX=index3
```

You can also use the &INDEX substitution parameter to create unique data set names for multiple indexes to be built. The substituted value for &INDEX is the name of the index being processed by Fast Path Indexer/EP when the mask of &INDEX is used with the DSNAME or OUTPUT_DSN_MASK keywords on the IX subcommand.

Since the INDEX keyword is not specified, all indexes that are defined for the database are processed. Four output data sets are generated, one for each index (for example, PFXIDX1, PFXIDX2, PFXIDX3, and PFXIDX4), each having a unique data set name. The example in Figure 65 shows how to build multiple secondary index databases to source a database by using the &INDEX parameter.

**Figure 65  Building multiple secondary index databases to a primary DEDB by using &INDEX**

```
BUILD DBD=PFXSRCDB
  IX OUTPUT_DSN_MASK=’IMS.&DBD.&INDEX’, DISP=SHR
```

Index partitioning

You can build partitioned indexes by using specific command syntax with the BUILD command. PFXLIB contains information for the partitioned index so that Fast Path Indexer/EP can determine the index partitions that are defined to a particular index. The Fast Path Indexer/EP BUILD command determines the index partitions that are defined to a particular index from the control block in the PFXLIB (for a PFX index) or ACBLIB (for an IBM native index). Figure 66 illustrates a partitioned PFX SHISAM or HISAM index.
You can include 500 partitions in a PFX partitioned index and 101 partitions in an IBM native index. A non-partitioned SHISAM or HISAM index data set has a 4-GB limit. If you use partitioning, however, you can include as many as 500 partitions, each containing 4 GB of space. Each of these partitions must mirror the original SHISAM or HISAM index design. Partitions let you use the SHISAM or HISAM structure while including much more information than can fit into one SHISAM or HISAM index data set.

When you execute the BUILD command, it builds all partitions; Fast Path Indexer/EP no longer supports building specific partitions.

**NOTE**
You must write your own exit routine to tell Fast Path Indexer/EP how to distribute the index data into the partitions.

IBM provides a sample partition exit in member DBFPSE00 in the IMS.SDFSSRC library.

When you partition your Fast Path Indexer/EP SHISAM or HISAM indexes, you instruct the indexes to distribute data by using the *vertical sequential* access method (default) or the *lateral sequential access* method. You indicate the access method in the PFX definition control block. For more information, see Chapter 6, “Primary DEDB and PFX index control block maintenance.” For information about IBM native index requirements, see the IBM documentation.

The following examples show how to select vertical or lateral processing for PFX SHISAM or HISAM indexes:

- **PFXCHILD macro**
  - Figure 16 on page 63 (macro example)
  - Figure 15 on page 63 (syntax diagram)
Performance considerations

You can improve performance when you are building an index by using one or more of the following techniques:

- input caching
- input thread processing
- index thread processing

Input caching

You can specify input caching of the primary DEDB or areas by using the ICACHE keyword. Input caching can enhance the processing of the index build function. Input caching of IOVF CIs can significantly enhance performance if the following conditions are present:

- The DEDB has a significant number of IOVF CIs that are likely to be processed by the function.
- Storage resources are available to support storage of all IOVF CIs.

**NOTE**

If you use a sequential input data set containing index information (created by the XSCAN utility) for input into the build process, you cannot use the ICACHE keyword to specify input caching of the primary DEDB or areas.

Input thread processing

For offline processing, the INPUT_THREADS keyword can be used to improve the performance of the build function. The INPUT_THREADS keyword lets you specify the maximum number of primary DEDB input areas to process concurrently. If resources are not available to support the specified number, parallel processing occurs only on the number that resources can support.
The following considerations apply to using INPUT_THREADS as a means of enhancing performance for the build function:

- The default (maximum) value for INPUT_THREADS is determined automatically by the product based on the number of input areas, CPU processors, and other system resources.
- Specifying a number less than the automatic default might improve performance, but specifying a number greater than the default value will be ignored.

**Index thread processing**

The INDEX_THREADS keyword specifies the maximum number of threads to use for parallel processing of the secondary index databases. If resources are not available to support the specified number, parallel processing occurs only on the number that resources can support.

The following considerations apply to using INDEX_THREADS as a means of limiting resource consumption by the index maintenance function:

- The default (maximum) value for the INDEX_THREADS keyword is determined automatically by the product based on the number of indexes, number of CPU processors, and other system resources.
- Specifying a value for INDEX_THREADS that is less than the automatic default might improve performance, but specifying a number greater than the default will be ignored.
- Each thread requires a resource-intensive sort process. The INDEX_THREADS keyword can be used to reduce the number of threads, thereby reducing the number of sorts required to process all indexes specified in the BUILD command set.
- All indexes are opened at once, and all index thread resources must be available throughout index build processing. Sufficient resources, including storage and sort work space must be available to accommodate the specified number of threads.

**Customizing sort processing**

The SORT_OPTION keyword provides a convenient method for supplying optional sort tuning parameters to your site’s Sort utility. The parameter that you specify will be used to control the sorting of intermediate data records used during the index build process.
SORT_OPTION=DYNALLOC (the default) specifies that sort work space will be dynamically allocated according to your Sort utility’s installation defaults. For more information, see the reference manual for the sort product that is used at your site.

**Sorting index information for the build function**

This section discusses how to use the SORT keyword and the SORT_OPTION keyword to sort index information when using an XSCAN data set created by the XSCAN utility.

**Using the SORT keyword**

When used with the IX subcommand, the SORT keyword can be used to specify whether the index information records are to be sorted into index key sequence before executing the build function. The default, SORT=YES, automatically sorts the scanned index records into index key sequence. If SORT=NO is specified, the index records are not sorted into index key sequence.

The example in Figure 67 shows how to sort scanned index records into index key sequence when either the DDNAME or DSNAME is also specified on the IX subcommand. For more information, see “Allocating the XSCAN data set” on page 158.

**Using the SORT_OPTION keyword**

Use the SORT_OPTION keyword to specify a list of sort utility control options. The SORT_OPTION keyword can only be used with SORT=YES (default); it cannot be used if SORT=NO is specified. The YES parameter provides a list of sort utility options. For more information about sort utility control options, see the reference manual for the sort product that is used at your site.

The example in Figure 67 shows how to sort index information records into index key sequence when the DDNAME is also specified on the IX subcommand, and how to specify a list of sort utility control options. For more information, see “Allocating the XSCAN data set” on page 158.
Rebuilding indexes during database change

You can request that any or all of the indexes associated with a primary DEDB be rebuilt simultaneously with the execution of the change function against the primary DEDB. The change function is performed by the Fast Path Reorg/EP product.

Specify the IX subcommand and associated keywords for the each of the indexes to be rebuilt. For details about the DEDB change function, see the *Fast Path Offline Suite User Guide*.

The IX subcommand in Figure 68 requests that all indexes that are registered to the primary DEDB be rebuilt during the change process.

The INDEX_THREADS keyword might be specified on the CHANGE command to specify the number of index threads to be used to rebuild indexes. Figure 69 requests that five threads be used for rebuilding all indexes that are registered to the primary DEDB during the change process.

For more information about using the INDEX_THREADS keyword, see “Index thread processing” on page 163.

**NOTE**
When rebuilding indexes during a database change, you cannot use the XSCAN data set that is generated by the XSCAN utility as input into the index build process.
Rebuilding indexes during database reload

You can request that any or all of the indexes associated with a primary DEDB be rebuilt simultaneously with the execution of the reload function against the primary DEDB. The reload function is performed by the Fast Path Reorg/EP product.

Specify the IX subcommand and associated keywords for the indexes to be rebuilt. For details about the DEDB reload function, see the Fast Path Offline Suite User Guide.

The IX subcommand in Figure 70 requests that all indexes registered to the DEDB be rebuilt during the reload process.

**Figure 70  Requesting index rebuild during DEDB reload**

```
RELOAD DBD=dbdname, IAREA=ALL, OAREA=ALL
   IX
```

The INDEX_THREADS keyword might be specified on the RELOAD command to specify the number of index threads to be used to rebuild indexes. Figure 71 requests that five threads be used for rebuilding all indexes that are registered to the primary DEDB during the reload process.

**Figure 71  Requesting index rebuild during DEDB reload**

```
RELOAD DBD=dbdname, IAREA=ALL, INDEX_THREADS=5
   IX
```

For more information about using the INDEX_THREADS keyword, see page 163.

---

**NOTE**

When rebuilding indexes during a database reload, you cannot use the XSCAN data set that is generated by the XSCAN utility as input into the index build process.

---

Sample BUILD command scenarios

The scenarios in this section illustrate how to use the BUILD command.
Scenario 1: Build of PFX indexes defined to a DEDB by using XSCAN input

In the example shown in Figure 72, the primary DEDB DIVSSD is input into a prior XSCAN step that created four separate data sets containing index information. This XSCAN-created index information will be used as input into the build process. Four indexes are defined for the primary DEDB: two are in SHISAM format and two are in HISAM format.

**NOTE**
The IAREA keyword is invalid when XSCAN input is used on the BUILD command.

Because the INDEX keyword is not specified on the IX subcommand, all four indexes that are defined for DIVSSD will be rebuilt from the four output data sets that are created by the XSCAN process.

The DSNAME keyword identifies the sequential input data set where the corresponding index information records are read. In the example in Figure 72, the &DBD and &INDEX variables represent the names of the DBD and index.

Because the indexes must be empty, an IDCAMS DELETE/DEFINE must be performed before executing the BUILD command.

**Figure 72  Build of PFX indexes defined to a DEDB by using XSCAN input**

```plaintext
//****************************************************
//*IDCAMS <-- for index 1, 2, 3, and 4
//*
//****************************************************
//BUILD  EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// DD DISP=SHR,DSN=IMSVS.RESLIB
//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
//PFXLIB DD DISP=SHR,DSN=BMC.PFXLIB
//DIVSI1 DD DISP=OLD,DSN=BMC.DIVSI1.DIVSI1
//DIVSI2 DD DISP=OLD,DSN=BMC.DIVSI2.DIVSI2
//DIVSI3 DD DISP=OLD,DSN=BMC.DIVSI3.DIVSI3
//DIVSI4 DD DISP=OLD,DSN=BMC.DIVSI4.DIVSI4
//PFPPRINT DD SYSOUT=*  
//PFPRPTS DD SYSOUT=* 
//PFPSYSIN DD *
GLOBAL DBRC=YES
BUILD DBD=DIVSSD
  IX DSNAME='BMC.&DBD.&INDEX.XSCAN',DISP=SHR
```
Scenario 2: Build for specified indexes defined to a DEDB by using XSCAN input

In the example in Figure 73, the primary DEDB EIVSSD is input into a prior XSCAN step, which created an XSCAN index data set for the EIVSI3 and EIVSI5 indexes. The XSCAN-created index information will be used as input into the build process. Six indexes are defined for the primary DEDB.

An IX subcommand is needed for each index to be rebuilt. The indexes will be processed simultaneously.

The INDEX keyword on the IX subcommand specifies the name of the index. The DSNAME keyword identifies the sequential input data set where the corresponding index information records were written by the XSCAN process.

Because indexes EIVSI3 and EIVSI5 must be empty, an IDCAMS DELETE/DEFINE must be performed before executing the BUILD command.

Figure 73  Build for specified indexes defined to a DEDB using XSCAN input

```hll
//****************************************************
//*IDCAMS <-- for index EIVSI3 and EIVSI5
//* //****************************************************
//BUILD     EXEC PGM=PFPMAIN,REGION=OM
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
//PFXLIB DD DISP=SHR,DSN=BMC.PFXLIB
//EIVSSD1 DD DISP=SHR,DSN=BMC.EIVSSD.EIVSSD1
//EIVSSD2 DD DISP=SHR,DSN=BMC.EIVSSD.EIVSSD2
//EIVSSD3 DD DISP=SHR,DSN=BMC.EIVSSD.EIVSSD3
//EIVSI3 DD DISP=OLD,DSN=BMC.EIVSI3.EIVSI3
//EIVSI5 DD DISP=OLD,DSN=BMC.EIVSI5.EIVSI5
//PFPPRINT DD SYSOUT=*  //PFPRPTS DD SYSOUT=*  //PFPSYSIN DD *
BUILD DBD=EIVSSD,IAREA=ALL
    IX INDEX=EIVSI3,
        DSNAME='BMC.&DBD.&INDEX.XSCAN',DISP=SHR
    IX INDEX=EIVSI5,
        DSNAME='BMC.&DBD.&INDEX.XSCAN',DISP=SHR
```
Scenario 3: Build of specified indexes defined to a DEDB by using primary DEDB input

In the example shown in Figure 74, the database COURSE is used as source input. Four indexes are defined for the primary DEDB: two are in HISAM format and two are in SHISAM format. Because the default (ALL) is used for the IAREA keyword on the BUILD command, all areas will be processed.

Because the indexes must be empty, an IDCAMS DELETE/DEFINE must be performed before executing the BUILD command.

Figure 74  Build of specified indexes defined to a DEDB by using primary DEDB input

```lang-none
//******************************************************
//*IDCAMS <-- idcsm for index 1, 2, 3, and 4
//* //******************************************************
//BUILD    EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB   DD DISP=SHR,DSN=BMC.PFP.LOAD
//          DD DISP=SHR,DSN=IMSVS.RESLIB
//IMSACB    DD DISP=SHR,DSN=IMSVS.ACBLIB
//PFXLIB    DD DISP=SHR,DSN=BMC.PFXLIB
//DIVSSD1   DD DISP=SHR,DSN=BMC.COURSE.DIVSSD1
//DIVSI1    DD DISP=OLD,DSN=BMC.COURSE.DIVSI1
//DIVSI2    DD DISP=OLD,DSN=BMC.COURSE.DIVSI2
//DIVSI3    DD DISP=OLD,DSN=BMC.COURSE.DIVSI3
//DIVSI4    DD DISP=OLD,DSN=BMC.COURSE.DIVSI4
//PFPPRINT  DD SYSOUT=* 
//PFPRPTS   DD SYSOUT=* 
//PFPSYSIN  DD *
//BUILD    DBD=COURSE
IX INDEX=DIVSI1
IX INDEX=DIVSI2
IX INDEX=DIVSI3
IX INDEX=DIVSI4
```

Scenario 4: Index build by using dynamic allocation of primary DEDB (INPUT_DSN_MASK) and indexes (OUTPUT_DSN_MASK)

In the example shown in Figure 75, the primary DEDB COURSE is used as input to the build utility. Four indexes are defined for the primary DEDB: two are in HISAM format and two are in SHISAM format.

The INPUT_DSN_MASK keyword dynamically allocates the input primary DEDB, and the OUTPUT_DSN_MASK keyword dynamically allocates the output index data sets that are rebuilt.
Scenario 5: Build of IBM native indexes defined to a DEDB by using XSCAN input

Because the indexes must be empty, an IDCAMS DELETE/DEFINE must be performed before executing the BUILD command.

Figure 75  Index build by using dynamic allocation of primary DEDB and indexes

```
//******************************************************
//*IDCAMS <-- for index 1, 2, 3, and 4
//* //******************************************************
//BUILD EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// DD DISP=SHR,DSN=IMSVS.RESLIB
//IMSACB DD DISP=SHR,DSN=IMSVS.ACBLIB
//PFXLIB DD DISP=SHR,DSN=BMC.PFXLIB
//PFPPRINT DD SYSOUT=* 
//PFPRPTS DD SYSOUT=* 
//PFPSYSIN DD *
BUild DBD=COURSE,IAREA=ALL, 
 INPUT_DSN_MASK='BMC.&DBD.&AREA'
 IX INDEX=(DIVSI1),
 OUTPUT_DSN_MASK='BMC.&DBD.&INDEX'
 IX INDEX=(DIVSI3),
 OUTPUT_DSN_MASK='BMC.&DBD.&INDEX'
 IX INDEX=(DIVSI2),
 OUTPUT_DSN_MASK='BMC.&DBD.&INDEX'
 IX INDEX=(DIVSI4),
 OUTPUT_DSN_MASK='BMC.&DBD.&INDEX'
```

Scenario 5: Build of IBM native indexes defined to a DEDB by using XSCAN input

In the example shown in Figure 72, the primary DEDB DIVSSD is input into a prior XSCAN step that created four separate data sets containing index information. This XSCAN-created index information will be used as input into the build process. Four indexes are defined for the primary DEDB: two are in SHISAM format and two are in HISAM format.

**NOTE**

The IAREA keyword is invalid when XSCAN input is used on the BUILD command.

Because the INDEX keyword is not specified on the IX subcommand, all four indexes that are defined for DIVSSD will be rebuilt from the four output data sets that are created by the XSCAN process.
Scenario 5: Build of IBM native indexes defined to a DEDB by using XSCAN input

The DSNAME keyword identifies the sequential input data set where the corresponding index information records are read. In the example in Figure 76, the &DBD and &INDEX variables represent the names of the DBD and index.

Because the indexes must be empty, an IDCAMS DELETE/DEFINE must be performed before executing the BUILD command.

Figure 76  Build of IBM native indexes defined to a DEDB by using XSCAN input

```
//******************************************************************************
//*IDCAMS <-- for index 1, 2, 3, and 4
//*
//******************************************************************************
//BUILD     EXEC PGM=PFPMAIN,REGION=0M
//STEPLIB DD DISP=SHR, DSN=BMC.PFP.LOAD
// DD DISP=SHR, DSN=IMSVS.RESLIB
//IMSACB DD DISP=SHR, DSN=IMSVS.ACBLIB
//DIVSI1 DD DISP=OLD, DSN=BMC.DIVSI1.DIVSI1
//DIVSI2 DD DISP=OLD, DSN=BMC.DIVSI2.DIVSI2
//DIVSI3 DD DISP=OLD, DSN=BMC.DIVSI3.DIVSI3
//DIVSI4 DD DISP=OLD, DSN=BMC.DIVSI4.DIVSI4
//PFPPRINT DD SYSOUT=*  
//PFPRTS DD SYSOUT=*  
//PFPSYSIN DD *
GLOBAL DBRC=YES
BUILD DBD=DIVSSD
 IX DSNAM='BMC.&DBD.&INDEX.XSCAN', DISP=SHR
```
This chapter provides information about the capabilities and use of the Fast Path Indexer/EP verify function. This function is executed by a command-driven utility that lets you verify the contents of an index to a Fast Path database following any event that might have caused an out-of-sync condition.

This chapter also discusses the capabilities and use of the Fast Path Indexer/EP index resynchronize function. This function is executed by a command-driven utility that lets you resynchronize an index with its associated Fast Path primary DEDB following any changes made to the primary DEDB.

**NOTE**

All topics in this chapter for the VERIFY command apply to both PFX and IBM native indexes. Topics for the RESYNC command only apply to PFX indexes. The RESYNC command cannot process IBM native indexes.

This chapter discusses the following topics:

- Introduction .......................................................... 174
- Verify function .................................................... 174
- Resynchronize function ........................................ 175
- Verify function inputs and outputs ............................. 175
- Resynchronize function inputs and outputs .................... 176
- DBRC considerations ............................................. 177
- Offline mode ....................................................... 177
- BMP mode .......................................................... 178
- Index validation command language ............................. 178
- Selecting the database and areas ................................. 180
  - Offline mode .................................................... 180
  - BMP mode ....................................................... 181
- Allocating the source area data set ......................... 182
- Specifying indexes to be verified or resynchronized ............ 182
  - Using the INDEX keyword or the &INDEX parameter .... 182
  - Using the DDNAME or DSNAME keyword .................. 183
- Processing the XSCAN data set ............................ 184
Introduction

Fast Path Indexer/EP provides two command-driven utilities that validate indexes: the verify function and the resynchronize function.

An XSCAN data set (created by the XSCAN utility) can be used in place of the DEDB area data set as input into these utilities. Only one of these data set types (XSCAN data set or DEDB area data set) can be used as input. For more information about using the XSCAN utility, see Chapter 8, “Index scan utility.”

Verify function

By using the command-driven index verify utility, you can validate the contents of an index to a primary DEDB offline or online in an IMS batch message processing (BMP) region. The VERIFY command lets you diagnose conditions that are causing an out-of-sync condition.

If the VERIFY command indicates mismatches between the primary DEDB and its associated index, the Fast Path Indexer/EP RESYNC command can be used to perform an online resynchronization of the secondary index database with its associated primary DEDB.
Resynchronize function

By using the command-driven index resynchronize utility, you can resynchronize an index to a primary DEDB without taking it offline. Unlike other online commands that run in an IFP region in Fast Path/EP Series products, the RESYNC command runs in an IMS BMP region and can be executed only while the database is online.

The RESYNC command performs a one-step verification and resynchronization of an index to its associated primary DEDB while the primary DEDB remains online to IMS.

**NOTE**
Before executing the RESYNC command in the specified PSB, you must ensure that the primary DEDB has been defined with at least read access, and that the secondary index database has been defined with read and write access in the PSB. For more information about creating the PSB, see “Creating the PSB for BMP processing” on page 135.

The RESYNC command can process only PFX indexes. It cannot process IBM native indexes.

Verify function inputs and outputs

The VERIFY command can accept input from an entire DEDB, from specific areas within a DEDB, or from an XSCAN data set created by the XSCAN function. Because the VERIFY command is essentially a read-only process, the only output from the process is a summary report. Inputs and outputs are shown in Figure 77.
Fast Path Indexer/EP generates a report containing a summary of statistics when the verify activities are complete. For an example of the Verify Summary Report, see the PFPVER member in the REPORTS data set.

**Resynchronize function inputs and outputs**

The RESYNC command can accept input from an entire DEDB, from specific areas within a DEDB, or from an XSCAN data set created by the XSCAN function. The output from the RESYNC command is one or more indexes that are associated with the primary DEDB. Inputs and outputs are shown in Figure 78.
Fast Path Indexer/EP generates a report containing a summary of statistics when the resynchronize activities are complete. For an example of the Resynchronization Summary Report, see the PFPRSC member in the REPORTS data set.

**DBRC considerations**

This section discusses DBRC considerations for the index verify and resynchronize functions.

**Offline mode**

*NOTE*

You *cannot* run the resynchronize function in offline mode.

When the index verify function is executed in offline mode and DBRC is active during execution, Fast Path Indexer/EP obtains authorization for “read with integrity” (RD) from DBRC for the database and each registered index. These conditions are shown in Table 22.
Multiple area data sets (MADS) are not supported for the source input areas. When executing an offline index verify, Fast Path Indexer/EP searches the ADS (area data set) list registered for each area (in collating sequence by DD name). The product selects the first ADS that is marked as available for use and that has no error queue elements (EQEs). If an ADS is found that meets both of these criteria, it is the only ADS that will be verified by Fast Path Indexer/EP. All other area data sets are ignored. If no area data set is marked as available, or if all available area data sets contain one or more EQEs, the verify function cannot be performed.

MADS are not supported for an index that uses a DEDB structure. When executing a verify for this type of index, Fast Path Indexer/EP searches the ADS (area data set) list registered for each area (in collating sequence by DD name). The product selects the first ADS that is marked as available for use and that has no EQEs. If an ADS is found that meets both of these criteria, it is the only ADS that will be verified by Fast Path Indexer/EP. All other area data sets are ignored. If no area data set is marked as available, or if all available area data sets contain one or more EQEs, the verify function cannot be performed.

### BMP mode

When the index verify or resynchronize process is executed, all databases must be online and available for input processing.

### Index validation command language

The keywords and subcommands that are available for the VERIFY command are listed in Table 23.

<table>
<thead>
<tr>
<th>Database</th>
<th>Authorization</th>
<th>Recovery Needed flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>source input areas</td>
<td>read with integrity (RD)</td>
<td>OFF</td>
</tr>
<tr>
<td>input index</td>
<td>read with integrity (RD)</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Table 23  VERIFY command keywords and subcommands

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the primary DEDB and areas (for input to the index verify process)</td>
<td>VERIFY</td>
<td>DBD IAREA</td>
</tr>
<tr>
<td>processing the XSCAN data set (for input to the index verify process)</td>
<td>IX</td>
<td>DDNAME DSNAME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Related keywords:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISP UNIT VOLSER</td>
</tr>
<tr>
<td>specifying indexes to be verified</td>
<td>IX</td>
<td>INDEX</td>
</tr>
<tr>
<td>changing the number of segments to be read before performing a checkpoint call</td>
<td>VERIFY</td>
<td>CHECKPOINT</td>
</tr>
<tr>
<td>enhancing performance of the verify process</td>
<td>VERIFY</td>
<td>INPUT_THREADS ICACHE INDEX_THREADS</td>
</tr>
<tr>
<td>customizing sort processing</td>
<td>VERIFY</td>
<td>SORT_OPTION</td>
</tr>
<tr>
<td>sorting index information</td>
<td>IX</td>
<td>SORT SORT_OPTION</td>
</tr>
</tbody>
</table>

The keywords and subcommands that are available for the RESYNC command are listed in Table 24.

Table 24  RESYNC command keywords and subcommands (part 1 of 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>selecting the database and areas (for input to the index resynchronize process)</td>
<td>RESYNC</td>
<td>DBD IAREA</td>
</tr>
<tr>
<td>processing the XSCAN data set (for input to the index resynchronize process)</td>
<td>IX</td>
<td>DDNAME DSNAME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Related keywords:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISP UNIT VOLSER</td>
</tr>
<tr>
<td>changing the number of segments to be read before performing a checkpoint call</td>
<td>RESYNC</td>
<td>CHECKPOINT</td>
</tr>
</tbody>
</table>
Selecting the database and areas

The DBD keyword identifies the name of the primary DEDB (DBD name) for which you want to verify or resynchronize an associated index or indexes. The DBD will be loaded from the corresponding member in the IMSACB library.

Offline mode

 NOTE

You cannot run the resynchronize function in offline mode.

In offline mode, if you want to verify indexes only for specific areas within the DEDB, use the IAREA keyword to specify the name of the area or areas to be used as input. You can specify one or more area names. If you omit the IAREA keyword, all areas defined in the DEDB are used as input to the process.

Areas might be specified on the IAREA keyword by using any combination of area names, area numbers, or area ranges. The following parameters are available for the IAREA keyword:

- IAREA=ALL (default) – Specify all areas of the DEDB.
- IAREA=areaname – Specify one or more areas by using the one-character to eight-character area name for each area specified. Multiple area names must be enclosed in parentheses and separated by commas.

Table 24  RESYNC command keywords and subcommands (part 2 of 2)

<table>
<thead>
<tr>
<th>Function</th>
<th>Command or subcommand</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>enhancing performance of the resynchronize process</td>
<td>RESYNC</td>
<td>ICACHE</td>
</tr>
<tr>
<td>specifying indexes to be resynchronized</td>
<td>IX</td>
<td>INDEX</td>
</tr>
<tr>
<td>sorting index information</td>
<td>IX</td>
<td>SORT SORT_OPTION</td>
</tr>
</tbody>
</table>

For syntax of the commands, subcommands, and keywords shown in Table 23 on page 179, see the Fast Path/EP Series Reference Manual.
IAREA=areanumber – Specify one or more areas by using the one-character to five-character area number for each area specified. Multiple area numbers must be enclosed in parentheses and separated by commas.

IAREA=(RANGE=(startarea,endarea)) – Use this syntax to specify a consecutive range of areas using either areaname or areanumber parameters. The area number associated with startarea must be less than the area number associated with endarea.

To specify all areas in a DEDB as input to the index verify process, use a command set like the example shown in Figure 79.

**Figure 79 Sample control statement for specifying all areas as input to the index verify**

```
VERIFY DBD=dbdname,IAREA=ALL
```

To specify only selected areas as input to the index verify process, use a command set like the example shown in Figure 80.

**Figure 80 Sample control statement for specifying selected areas as input to the index verify**

```
VERIFY DBD=dbdname,IAREA=(AREANAM1,AREANAM3,RANGE=(5,8))
```

This control statement uses a combination of area names and area numbers to request that the specified areas are to be input to the index verify process. The UOWs in areas named AREANAM1 and AREANAM3 will be input. Also, all consecutive areas from area number 5 to area number 8 (area5, area6, area7 and area8) will be input.

**BMP mode**

If you use a sequential input data set containing index information (created by the XSCAN utility) for input into the resynchronize process, you can specify the IAREA keyword. However, if a sequential data set containing index information is not used for input, the IAREA keyword must *not* be specified. To limit the areas to be processed, use the DFSCTL DD statement to specify the appropriate SETR command. An example is shown in Figure 81.

**Figure 81 Limiting areas to be processed for an index verify or resynchronize in BMP mode**

```
//DFSCTL DD *
SETR DB=dbdname,PCB=label,AL=(area1,area2,...,arean)
```
Allocating the source area data set

For online verify and resynchronization processing, the area data set is accessed by using IMS Control Region Services. No JCL is needed for the area data sets; if JCL exists, it is ignored.

### Specifying indexes to be verified or resynchronized

Use the IX subcommand with the VERIFY command or the RESYNC command to specify which indexes are to be verified or resynchronized. At least one IX subcommand is required. By default, all indexes that are registered to the primary DEDB are verified or resynchronized. The example in Figure 82 shows how to use one IX subcommand to verify all indexes for the primary DEDB.

**Figure 82  Verifying all indexes for the primary DEDB**

```
VERIFY  DBD=dbdname,IArea=ALL
       IX
```

The example in Figure 83 shows how to use one IX subcommand to resynchronize all indexes for the primary DEDB.

**Figure 83  Resynchronizing all indexes for the primary DEDB**

```
RESYNC DBD=dbdname
       IX
```

### Using the INDEX keyword or the &INDEX parameter

You can use the INDEX keyword to specify a single index or multiple indexes to be verified or resynchronized. Use one IX subcommand with the INDEX keyword for each index to be verified or resynchronized. The example in Figure 84 shows how to verify an individual index for the primary DEDB.
Using the DDNAME or DSNAME keyword

Either the DDNAME or the DSNAME keyword must also be specified on the IX subcommand to specify where the XSCAN index information records are read. For more information about using the DDNAME keyword or the DSNAME keyword to process the XSCAN data set, see “Processing the XSCAN data set” on page 184.
**Processing the XSCAN data set**

The XSCAN data set that is created by the XSCAN utility can be used for input into the build, verify, and resynchronize functions.

When processing the verify or resynchronize function, the XSCAN data set can be supplied in the JCL, or it can be accessed by using dynamic allocation. If you want to use JCL to supply the XSCAN data set, specify the DDNAME keyword with the IX subcommand. If you want the XSCAN data set to be dynamically allocated, specify the DSNAME keyword with the IX subcommand. If the DDNAME keyword or the DSNAME keyword is not specified, the default scan of the primary DEDB is used to create the required index information records.

For dynamic allocation, you can use the DISP keyword to control the allocation and disposition of the data set. The DISP=USE status parameter indicates conditional allocation. If the data set does not exist, it is created (as if DISP=NEW had been specified). If the data set already exists, it is reallocated (as if DISP=OLD had been specified). If CATLG is specified, the normal and conditional parameters are changed from CATLG to KEEP. In addition to the DISP keyword, the UNIT and VOLSER keywords can also be used to control the allocation of a data set. For more information about these keywords, see the *Fast Path/EP Series Reference Manual*.

**Allocating secondary index databases for the verify function**

When the product allocates the secondary index databases for the index verify function, the product accessed the secondary index databases by using IMS Control Region Services. No JCL is needed for the area data sets; if JCL exists, it is ignored.

**Changing the frequency of checkpoint calls**

Fast Path Indexer/EP automatically uses a value of 10,000 segment reads before performing a symbolic checkpoint call. You can increase or decrease this value by specifying a numeric value on the CHECKPOINT keyword.

**NOTE**

The CHECKPOINT keyword functions in BMP mode only. If you specify a value for CHECKPOINT with an offline command, the keyword will be ignored.
In the example shown in Figure 88, Fast Path Indexer/EP will perform 15,000 segment reads before performing a symbolic checkpoint call during an online resynchronize of the index that is specified on the IX subcommand.

![Figure 88 Specifying checkpoint call frequency during an online resync](RESYNC_DBD=dbdname,CHECKPOINT=15000)

**Performance considerations for the verify function**

The performance considerations discussed in the following sections pertain to the verify function only. These considerations do not apply to the resynchronize function.

**Offline mode**

When running an index verify in offline mode, you can use input caching, limit the number of threads that are processed, or combine both of these techniques to improve performance.

**Input Caching**

For offline processing, you can specify input caching of the primary DEDB or areas by using the ICACHE keyword. Input caching can enhance the processing of the index verify function. Input caching of IOVF CIs can significantly enhance performance if the following conditions are present:

- The DEDB has a significant number of IOVF CIs that are likely to be processed by the function.
- Storage resources are available to support storage of all IOVF CIs.

**Input Thread Processing**

For offline processing, the INPUT_THREADS keyword can be used to improve the performance of the verify function. The INPUT_THREADS keyword lets you specify the maximum number of input areas to process concurrently. If resources are not available to support the specified number, parallel processing occurs only on the number that resources can support.
The following considerations apply to using INPUT_THREADS as a means of enhancing performance for the verify function:

- The default (maximum) value for INPUT_THREADS is determined automatically by the product based on the number of input areas, CPU processors, and other system resources.

- Specifying a number less than the automatic default might improve performance, but specifying a number greater than the default value will be ignored.

**Index Thread Processing**

The INDEX_THREADS keyword specifies the maximum number of threads to use for parallel processing of the secondary index databases. If resources are not available to support the specified number, parallel processing occurs only on the number that resources can support.

The following considerations apply to using INDEX_THREADS as a means of limiting resource consumption by the index maintenance function:

- The default (maximum) value for the INDEX_THREADS keyword is determined automatically by the product based on the number of indexes, number of CPU processors, and other system resources.

- Specifying a value for INDEX_THREADS that is less than the automatic default might improve performance, but specifying a number greater than the default will be ignored.

- Each thread requires a resource-intensive sort process. The INDEX_THREADS keyword can be used to reduce the number of threads, thereby reducing the number of sorts required to process all indexes specified in the VERIFY command set.

- All indexes are opened at once, and all index thread resources must be available throughout index build processing. Sufficient resources, including storage and sort work space, must be available to accommodate the number specified.

**BMP mode**

When executing an index verify in BMP mode, Fast Path Indexer/EP uses IMS services to read the primary and secondary index databases. As with any application BMP, the execution time is governed by the speed of IMS calls and logging services. Verifying an index in BMP mode will require much longer to run than verifying an index in offline mode.
Customizing sort processing for the verify function

The SORTOPTION keyword provides a convenient method for supplying optional sort tuning parameters to your site’s Sort utility. The parameter that you specify will be used to control the sorting of intermediate data records that are used during the verification process.

SORTOPTION=DYNALLOC (the default) specifies that sort work space will be dynamically allocated according to your Sort utility’s installation defaults.

For more information, see the reference manual for the sort product used at your site.

Sorting index information for the verify and resynchronize functions

This section discusses how to use the SORT keyword and the SORTOPTION keyword to sort index information when using an XSCAN data set created by the XSCAN utility.

Using the SORT keyword

When used with the IX subcommand, the SORT keyword can be used to specify whether the index information records are to be sorted into index key sequence before executing verify and resynchronize functions. The default, SORT=YES, automatically sorts the scanned index records into index key sequence. SORT=NO does not sort the records.

Using the SORTOPTION keyword

Use the SORTOPTION keyword to specify a list of sort utility control options. The SORTOPTION keyword can only be used with SORT=YES (default); it cannot be used if SORT=NO is specified. The YES parameter provides a list of sort utility options. For more information about sort utility control options, see the reference manual for the sort product that is used at your site.
The example in Figure 89 shows how to sort index information records into index key sequence when the DDNAME is specified on the IX subcommand, and how to specify a list of sort utility control options. For more information, see “Allocating the XSCAN data set” on page 158.

Figure 89  Sorting index information by using SORT and SORT_OPTION keywords

RESYNC  DBD=dbdname, IAREA=ALL
IX  INDEX=index1
    [DDNAME=ddname,]
    [SORT=YES|NO,] [SORT_OPTION=(option,...)]

Sample VERIFY command scenarios

The scenarios in this section illustrate how to use the VERIFY command.

Scenario 1: Verify all indexes defined to a DEDB by using primary DEDB input

In the example shown in Figure 90, the database COURSIIBM is used as source input. Four IBM native indexes are defined for the primary DEDB: two are in HISAM format, and two are in SHISAM format. Because the INDEX keyword is not specified, all defined indexes will be verified. The PFXLIBA and PFXLIBB DD statements are not needed when processing IBM native indexes.

Figure 90  Verify all indexes defined to a DEDB by using primary DEDB input

//VERIFY EXEC PGM=DFSRRC00,REGION=0M, //       PARM=(BMP,PFPMAIN,psbname) //IMSACBA DD DISP=SHR,DSN=IMSVS.ACBLIBA //IMSACBB DD DISP=SHR,DSN=IMSVS.ACBLIBB //PFPPRINT DD SYSOUT=* //STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD // DD DISP=SHR,DSN=IMSVS.RESLIB //MODSTAT DD DISP=SHR,DSN=IMSVS.MODSTAT //PPRPTS DD SYSOUT=* //PFPSYSIN DD * //VERIFY DBD=COURSEIBM IX
Scenario 2: Verify specified indexes defined to a DEDB by using XSCAN input

In the example in Figure 91, the primary DEDB EIVSSD was input into a prior XSCAN step, which created an XSCAN index data set for EIVSI3 and EIVSI5 indexes. The XSCAN-created index information will be used as input to the verify process. Six indexes are defined for the primary DEDB.

An IX subcommand is needed for each index to be rebuilt. The indexes will be processed simultaneously.

The INDEX keyword on the IX subcommand specifies the name of the index. The DSNAME keyword identifies the sequential input data set where the corresponding index information records were written by the XSCAN process.

Because indexes EIVSI3 and EIVSI5 must be empty, an IDCAMS DELETE/DEFINE must be preformed before executing the VERIFY command.

Figure 91  Verify specified indexes defined to a DEDB by using XSCAN input

```plaintext
//VERIFY EXEC PGM=DFSRRC00,REGION=0M,
  PARM=(BMP,PFPMAIN,psbname)
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
//MODSTAT DD DISP=SHR,DSN=IMSVS.MODSTAT
//IMSACBA DD DISP=SHR,DSN=IMSVS.ACBLIBA
//IMSACBB DD DISP=SHR,DSN=IMSVS.ACBLIBB
//PFXLIBA DD DISP=SHR,DSN=BMC.PFXLIBA
//PFXLIBB DD DISP=SHR,DSN=BMC.PFXLIBB
//PFPPRINT DD SYSOUT=*
//PFPRPTS DD SYSOUT=*  
//PFPSYSIN DD *
VERIFY DBD=EIVSSD,IAREA=ALL
  IX INDEX=EIVSI3,
     DSNAME='BMC.&DBD.&INDEX.XSCAN',DISP=SHR
  IX INDEX=EIVSI5,
     DSNAME='BMC.&DBD.&INDEX.XSCAN',DISP=SHR
```
Sample RESYNC command scenarios

The scenarios in this section illustrate how to use the RESYNC command.

Scenario 1: Resynchronize all indexes defined to a DEDB by using primary DEDB input

In the example shown in Figure 92, the database COURSE is used as source input. Four indexes are defined for the primary DEDB: two are in HISAM format, and two are in SHISAM format. Because the INDEX keyword is not specified, all defined indexes will be resynchronized.

Figure 92  Resynchronize all indexes defined to a DEDB by using primary DEDB input

```
//RESYNC EXEC PGM=DFSRRC00,REGION=0M,
  PARM=(BMP,PFPMAIN,psbname)
//STEPLIB DD DISP=SHR,DSN=BMC.PFP.LOAD
// MODSTAT DD DISP=SHR,DSN=IMSVS.MODSTAT
//IMSACBA DD DISP=SHR,DSN=IMSVS.ACBLIBA
//IMSACBB DD DISP=SHR,DSN=IMSVS.ACBLIBB
//PFXLIBA DD DISP=SHR,DSN=BMC.PFXLIBA
//PFXLIBB DD DISP=SHR,DSN=BMC.PFXLIBB
//PFPPRINT DD SYSOUT=*  //PFPRPTS DD SYSOUT=*  //PFPSYSIN DD *
RESYNC DBD=COURSE  
IX
```

Scenario 2: Resynchronize specified indexes defined to a DEDB by using XSCAN input

In the example in Figure 93, the primary DEDB EIVSSD was input into a prior XSCAN step, which created an XSCAN index data set for EIVSI3 and EIVSI5 indexes. The XSCAN-created index information will be used as input to the verify process. Six indexes are defined for the primary DEDB.

An IX subcommand is needed for each index to be rebuilt. The indexes will be processed simultaneously.

The INDEX keyword on the IX subcommand specifies the name of the index. The DSNAME keyword identifies the sequential input data set where the corresponding index information records were written by the XSCAN process.
### Scenario 2: Resynchronize specified indexes defined to a DEDB by using XSCAN input

**Figure 93**  Resynchronize specified indexes defined to a DEDB by using XSCAN input

```
//RESYNC  EXEC PGM=DFSRRC00,REGION=0M,
  //       PARM=(BMP,PFPMAIN,psbname)
//STEPLIB  DD DISP=SHR,DSN=BMC.PFP.LOAD
//         DD DISP=SHR,DSN=IMSVS.RESLIB
//MODSTAT  DD DISP=SHR,DSN=IMSVS.MODSTAT
//IMSAACBA DD DISP=SHR,DSN=IMSVS.ACBLIBA
//IMSAACBB DD DISP=SHR,DSN=IMSVS.ACBLIBB
//PXLLIBA  DD DISP=SHR,DSN=BMC.PFLLIBA
//PXLLIBB  DD DISP=SHR,DSN=BMC.PFLLIBB
//PFPPRINT DD SYSOUT=*  
//PFPRPTS  DD SYSOUT=* 
//PFPSYSIN DD *  
RESYNC  DBD=EIVSSD,IAREA=ALL
  IX  INDEX=EIVS13,   
       DSNAME='BMC.&DBD.&INDEX.XSCAN',DISP=SHR
  IX  INDEX=EIVS15,   
       DSNAME='BMC.&DBD.&INDEX.XSCAN',DISP=SHR
```
Scenario 2: Resynchronize specified indexes defined to a DEDB by using XSCAN input
Index registration and verification utility

This chapter describes the Fast Path Indexer/EP Registration Verification (PFXLIST) utility. The PFXLIST utility can be used to verify index registrations and to create source statement definitions.

This chapter discusses the following topics:

- PFXLIST overview ................................................................. 193
- LIST functions ................................................................. 194
- BUILD functions .............................................................. 194
- Inputs and outputs .............................................................. 194
- JCL requirements ............................................................... 195
- Control statements ............................................................ 197
- Examples ............................................................................ 198
- PFXLIST sample report ....................................................... 199

PFXLIST overview

The Registration Verification (PFXLIST) utility is provided with Fast Path Indexer/EP as an administrative and diagnostic service aid. PFXLIST performs two main functions: LIST and BUILD. Each function is implemented using a unique utility control statement on PFXSYSIN.

NOTE

The PFXLIST utility can only process PFX indexes. It cannot process IBM native indexes.
LIST functions

When you specify the LIST utility control statement, PFXLIST performs the following functions:

- verifies PFXLIB against ACBLIB

  PFXLIST verifies that the registration control blocks in PFXLIB match the IMS ACBs for a specified primary DEDB and all of its secondary index databases.

- reports on registration control blocks

  PFXLIST generates a report that includes registration data for a specified primary DEDB and all of its secondary index databases. This information is delineated by PSB information and DBD information.

BUILD functions

When you specify the BUILD utility control statement, PFXLIST performs the following functions:

- verifies PFXLIB against ACBLIB

  PFXLIST verifies that the registration control blocks in PFXLIB match the IMS ACBs for a specified primary DEDB and its secondary index databases.

- disassembles DBDs and creates DBD source statements

  PFXLIST creates the DBD source statements for registered databases.

- disassembles PSBs and creates PSB source statements

  PFXLIST creates the PSB source statements for registered PSBs. The source statement build function is supported only for control blocks that are required for index registration.

Inputs and outputs

Figure 94 illustrates the inputs and outputs of the PFXLIST utility.
Figure 94  PFXLIST inputs and outputs

If you request that PFXLIST report on a specific primary DEDB on the PFXSYSIN DD statement (using the LIST control statement), PFXLIST reports on all secondary index databases associated with that primary DEDB.

JCL requirements

Execute PFXLIST as a standard OS job. Figure 95 provides sample JCL that you can use to execute the PFXLIST reporting (LIST) function.

Figure 95  PFXLIST sample JCL

```plaintext
//PFXLIST  JOB (xxxx).....etc.
//LIST     EXEC PGM=PFXLIST,REGION=4096K
//STEPLIB   DD DSN=BMC.PFX.PGMLOAD,DISP=SHR
//         DD DSN=IMSVS.RESLIB,DISP=SHR
//PFXLIB    DD DSN=PFX.REGI.DSN,DISP=SHR
//         DD DSN=IMSVS.PSBLIB,DISP=SHR
//         DD DSN=IMSVS.DBDLIB,DISP=SHR
//         DD DSN=IMSVS.ACBLIB,DISP=SHR
//SYSPRINT  DD SYSOUT=*<=optional
//DBDSRC    DD DSN=PFX.DBDSRC,DISP=SHR <=optional
//PSBSRC    DD DSN=PFX.PSBSRC,DISP=SHR <=optional
//PFXSYSIN  DD *
LIST DBD=ALL
```

Descriptions for each statement that you can use in the PFXLIST JCL follow.
EXEC

Required. You must specify PGM=PFXLIST. The recommended region size is 4096 KB.

STEPLIB DD

Required. Defines the load library that is required to run PFXLIST.

PFXLIB DD

Required. Defines the input data set used to hold the registration control blocks. This data set must reside on DASD. Ensure that you specify the correct PFXLIBx data set (active or inactive).

IMS DD

Required. Defines the IMS PSB and DBD load libraries. The PSBLIB must appear before the DBDLIB.

IMSACB DD

Required. Defines the library that contains the DMB that describes the primary DEDB and its associated indexes.

SYSPRINT DD

Required. Defines the output of the reports generated by PFXLIST that are routed to the output stream. If you supply DCB information for this statement, specify RECFM=FBA and LRECL=121.

DBDSRC DD

Required if BUILD=DBD is specified. Defines the output PDS that is to be used for the DBD source statements, which are generated by PFXLIST. This DBD source can be used as input to the IMS DBDGEN utility to regenerate the DBDs for the primary DEDBs and secondary index databases. This DD statement is required to generate any DBD source statements.

PSBSRC DD

Required if BUILD=PSB is specified. Defines the output PDS that is to be used for the PSB source statements, which are generated by PFXLIST. This PSB source can be used as input to the IMS PSBGEN utility to regenerate the PSBs for the PFX-related databases. This DD statement is required to generate any PSB source statements.
PFXSYSIN DD

Required. Defines the input data set that contains the single control statement for PFXLIST. This data set can reside on DASD, or it can be routed through the input stream. If you supply DCB information for this statement, specify RECFM=FB and LRECL=80.

Control statements

This section describes the PFXLIST control statements and keywords. You can code control statements and keywords by using a free format in columns 1 through 71. If you begin specifying data for a keyword on one line and need to continue on to the next line, perform the following steps:

1 Include a closing parenthesis mark before column 71 on the current line.

2 Duplicate the same keyword for which you are still specifying data on the next line.

3 Specify the remaining data on the new line.

You can report, verify, and create DBD source statements by specifying control information about a PFXSYSIN DD and PSB statement. You can specify a particular DBD and/or PSB. You can request that PFXLIST report all registered DBDs or PSBs by specifying LIST DBD=ALL or LIST PSB=ALL on the PFXSYSIN DD statement.

Figure 96 shows samples of the available keywords and parameters for the LIST and BUILD control statements.

Figure 96  PFXLIST control statements and keywords

```plaintext
//PFXSYSIN DD *
LIST [DBD=xxxxxxxxx]
   =x
   =ALL
 [,PSB=xxxxxxxx]
   =ALL
   =x
   =(x,x,x...)  
BUILD [DBD=xxxxxxxxx]
   =ALL
   =x
   =(x,x,x...)  
 [,PSB=xxxxxxxxx]
   =ALL
   =x
   =(x,x,x...)  
```
Table 25 describes the control statements and keywords that are shown in Figure 96.

Table 25  PFXLIST control statements and keywords

<table>
<thead>
<tr>
<th>Control Statement</th>
<th>Keyword</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST</td>
<td></td>
<td></td>
<td>Required if you want to list information about registered DBDs or PSBs. This control statement generates a report that describes the PFX registration control blocks.</td>
</tr>
<tr>
<td>DBD</td>
<td></td>
<td>list information about registered DBDs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>list information for all registered DBDs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>list information for only this registered DBD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x,x,x,...)</td>
<td>list information for specific registered DBDs</td>
<td></td>
</tr>
<tr>
<td>PSB</td>
<td></td>
<td>list information about registered PSBs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>list information for all registered PSBs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>list information for only this registered PSB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x,x,x,...)</td>
<td>list information for specific registered PSBs</td>
<td></td>
</tr>
<tr>
<td>BUILD</td>
<td></td>
<td></td>
<td>Required if you want to create (build) DBD or PSB source statements</td>
</tr>
<tr>
<td>DBD</td>
<td></td>
<td>create source statements for a registered DBD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>create source statements for all registered DBDs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>create source statements for only this registered DBD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x,x,x,...)</td>
<td>create source statements for specific registered DBDs</td>
<td></td>
</tr>
<tr>
<td>PSB</td>
<td></td>
<td>create source statements for a registered PSB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>create source statements for all registered PSBs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>create source statements for only this registered PSB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x,x,x,...)</td>
<td>create source statements for specific registered PSBs</td>
<td></td>
</tr>
</tbody>
</table>

Examples

Using the databases shown in Appendix B, “Fast Path Indexer/EP sample application,” the following example illustrates how to list registration information for the primary DEDB and all of its associated indexes:

```
LIST DBD=DIVSSD
```

To list information about all registered PSBs, use the following syntax:

```
LIST PSB=ALL
```
To list information about all registered DBDs and PSBs, use the following syntax:

```
LIST DBD=ALL
LIST PSB=ALL
```

To build DBD and PSB source statements for all registered DBDs and PSBs, use the following syntax:

```
BUILD DBD=ALL
BUILD PSB=ALL
```

**PFXLIST sample report**

The Fast Path Indexer/EP Registration Verification report (Figure 97) lists information about primary DBDs, index DBDs, and index PSBs.

**Figure 97  Fast Path Indexer/EP Registration Verification report (part 1 of 2)**

```plaintext
SOURCE DEFINITION HEADER BLOCK
0000 8ED000F8 C4E2C9E2 C4C9E5E2 E2C44040 0093068F 0101014F 00200000 F0F361F0
0020 F961F9F3 F1F04BF0 F7000000 00050003 00020060 0000003C 00000000

SOURCE ENTRY DEFINITION BLOCK
0000 E2E3E4C4 C5D5E340 C3D6E4D9 E2C54040 00040003 00010001 0004000E 0B000001
0020 E2E3E4C4 E2D5D4C5 00080018 00000000 00000000 00000000 00000000 00000000
0040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0060 E4C4D5E4 D4C20001 00050000 61C3D240 00080018 00000000 00000000 00000000
0080 40400000 00041000 00000000 00000000 00000000 00000000 00000000 00000000
00A0 00000000 00010000 E2E3E4C4 C6D5D4C5 00210013 00000000 00000000 00000000
00C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00E0 00000000 D5000014 00000000 00000000 00000000 00000000 00000000 00000000
0100 C4E2C9D9 D6D6E340 E2D9C3E5 C1D3E440 40400404 40400000 40400000 40400000
0120 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

SOURCE SEGMENT NAME: STUDENT
TARGET SEGMENT NAME: COURSE
SOURCE SEGMENT CODE: 4
TARGET SEGMENT CODE: 1
INDEX TYPE: NON-DEDB
NUMBER OF SRCH FIELD: 1
  STUDSNME,START=9,BYTES=25
NUMBER OF SUBSEQ FIELDS: 3
STUDNUMB,START=2,BYTES=6
/CK,START=6,BYTES=4
/CK,START=1,BYTES=5
NUMBER OF DDATA FIELDS: 1
```
Figure 97  Fast Path Indexer/EP Registration Verification report (part 2 of 2)

<table>
<thead>
<tr>
<th>STUDFNME,START=34,BYTES=20</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX SEGMENT NAME: PFXROOT</td>
</tr>
<tr>
<td>INDEX SEQUENCE FIELD: SRCVALU</td>
</tr>
<tr>
<td>INDEX DBD NAME: DIVSI6</td>
</tr>
</tbody>
</table>

PFX INDEX DEFINITION HEADER BLOCK

```
0000 8ED00078 C4E2C9C9 C4C9E5E2 C9F64040 0093067F 0095219F 00200000 F0F361F0
0020 F561F9F3 F1F34BF4 F7000000 00010003 C4C9E5E2 E2C44040 0ED039EC 8ED039B0
0040 00000000 00000098 0000
```

HIERARCHICAL LEVEL OF TARGET SEGMENT: 1
HIERARCHICAL LEVEL OF SOURCE SEGMENT: 3

PFX INDEX ENTRY DEFINITION BLOCK...
Appendixes

This part presents the following topics:

Chapter A
  Diagnostic tools and procedures ............................................. 203

Chapter B
  Fast Path Indexer/EP sample application .................................... 215
Diagnostic tools and procedures

The tools, procedures, and information requests in this appendix were prepared by the BMC Customer Support staff to help you gather information necessary for them to provide timely and quality support.

This appendix discusses the following topics:

Diagnostic tools .......................................................... 203
  PFX Dump tool .......................................................... 204
  PFX Trace tool .......................................................... 204
Diagnostic procedures .................................................. 205
  Note software product levels ........................................... 205
  Locate generated SVC dump ........................................... 205
  Define the general problem ............................................ 207
  Define PFX registration problems ................................... 208
  Define index retrieval or maintenance problems ................. 209
  Provide duplication and resolution information ................... 209
Problem summary and documentation form .......................... 210

Diagnostic tools

Fast Path Indexer/EP includes two diagnostic tools. You should use these tools only when requested to do so by BMC Customer Support. Customer Support might ask you for detailed information that can be provided by these diagnostic tools. Output from either or both of these tools might be requested as part of the information that you must gather as discussed in this appendix.

This section discusses the following utilities:

  ■ PFX Dump tool

  Use this tool to create an SVC dump, which helps to debug problems that occur because of an abend.
PFX Dump tool

If the dependent control region abends while Fast Path Indexer/EP is in control, the product automatically generates an SVC dump. If the dependent control region abends while Fast Path Indexer/EP is not in control, but the PFXDUMP DD statement is present, the product will generate an SVC dump.

The PFX Dump tool provides information that can be used to debug problems caused by an abend. When instructed by your BMC Customer Support, use the PFXDUMP DD statement to initiate a dump.

Include the PFXDUMP DD statement in your JCL only when you can re-create an abend and want a copy of the SVC dump. The following is an example of the statement that you should use:

```assembly
//PFXDUMP DD DUMMY
RECFM=VB
LRECL=4096
BLKSIZE=5000
```

PFX Trace tool

The PFX Trace tool creates a sequential record of all PFX DL/I calls. When you activate this tool, it serves the dependent region in which its DD statement is included. When instructed by BMC Customer Support, use the PFXTRACE DD statement to initiate a trace.

You can write the output from this DD statement to a file, or you can print it. To perform either task, use the following DCB attributes:

```assembly
RECFM=VB
LRECL=4096
BLKSIZE=5000
```
To activate the PFX Trace tool, include the following DD statement in the dependent region JCL:

```plaintext
//PFXTRACE DD DSN=xxxxxxxxx,DCB=(RECFM=VB,
   LRECL=4096,BLKSIZE=5000)
```

Fast Path Indexer/EP accumulates the trace records until the PFXTRACE DD statement is removed from the JCL and the region is restarted.

## Diagnostic procedures

To assist BMC Customer Support in providing a faster response to any problem, you should have certain information available when you call. Information about software product levels and problem descriptions are required to resolve the problem.

The procedures, questions, and information requests in this appendix were prepared by BMC Customer Support to help you receive timely and quality support. Your initial effort will help resolve issues involving BMC products. You might want to copy and fill out the forms in this appendix and send them to BMC for further discussion.

### Note software product levels

Gather the following product level information:

- IBM IMS version, release, and PUT level
- IBM CICS version, release, and PUT level
- IBM z/OS version
- Fast Path Indexer/EP (PFX) version, release, and maintenance level

### Locate generated SVC dump

Fast Path Indexer/EP will automatically generate an SVC dump when a product abend occurs. Following the abend, MVS will generate a notification message that the SVC dump has been created. You should locate the SVC dump produced by the product abend. The data set name for the SVC dump can be found in the JES messages in the JOBLOG. BMC might request that you supply the SVC dump with other requested documentation.
If Fast Path Indexer/EP is unable to automatically generate an SVC dump following a product abend or if the PFX Dump utility is unable to create an SVC dump, the error message BMC110980E might be generated with return code 8. Table 26 lists explanations for reason codes that might be issued with the RC=8.

### Table 26  Reason codes for RC=8 on message BMC110980E (part 1 of 2)

<table>
<thead>
<tr>
<th>Reason code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No SVC dump was requested.</td>
</tr>
<tr>
<td>1</td>
<td>An SVC dump was successfully started.</td>
</tr>
<tr>
<td>2</td>
<td>An SVC dump was suppressed because the capture phase of another SVC dump was in progress.</td>
</tr>
<tr>
<td>3</td>
<td>An SVC dump was suppressed by a request by the installation (example: DUMP=NO at IPL or CHNGDUMP SET, NODUMP).</td>
</tr>
<tr>
<td>4</td>
<td>An SVC dump was suppressed by a SLIP NODUMP command.</td>
</tr>
<tr>
<td>5</td>
<td>An SVC dump was suppressed because a SYS1.DUMP data set was not available.</td>
</tr>
<tr>
<td>6</td>
<td>An SVC dump was suppressed because an I/O error occurred during initialization of the SYS1.DUMP data set.</td>
</tr>
<tr>
<td>8</td>
<td>An SVC dump was suppressed because an SRB could not be scheduled to activate the dump tasks in the requested address spaces.</td>
</tr>
<tr>
<td>9</td>
<td>An SVC dump was suppressed because a terminating error occurred in SVC dump before the first dump record was written.</td>
</tr>
<tr>
<td>A</td>
<td>An SVC dump was suppressed because a status stop SRB condition was detected.</td>
</tr>
<tr>
<td>B</td>
<td>An SVC dump was suppressed by Dump Analysis and Elimination (DAE).</td>
</tr>
<tr>
<td>15</td>
<td>The parameter list address is zero.</td>
</tr>
<tr>
<td>16</td>
<td>The parameter list is not a valid SVC or SNAP parameter list.</td>
</tr>
<tr>
<td>17</td>
<td>The caller-supplied data set is not supported.</td>
</tr>
<tr>
<td>18</td>
<td>The start address is greater than or equal to the end address in a storage list.</td>
</tr>
<tr>
<td>19</td>
<td>The caller-supplied header is longer than 100 characters.</td>
</tr>
<tr>
<td>1A</td>
<td>The caller requested a 4K buffer but did not reserve it.</td>
</tr>
<tr>
<td>1B</td>
<td>A storage list overlaps the 4K buffer.</td>
</tr>
<tr>
<td>1C</td>
<td>The caller-supplied DCB is not valid.</td>
</tr>
<tr>
<td>1E</td>
<td>An ASID in the ASID list is syntactically not valid.</td>
</tr>
<tr>
<td>22</td>
<td>The 4K buffer was requested with an SVC dump already in progress.</td>
</tr>
<tr>
<td>25</td>
<td>A subpool ID that was not valid was specified in the subpool list.</td>
</tr>
<tr>
<td>28</td>
<td>Part of the parameter list is inaccessible.</td>
</tr>
<tr>
<td>29</td>
<td>The caller-supplied DCB is inaccessible.</td>
</tr>
<tr>
<td>2A</td>
<td>The caller-supplied storage list is inaccessible.</td>
</tr>
<tr>
<td>2B</td>
<td>The caller-supplied header data is inaccessible.</td>
</tr>
<tr>
<td>2C</td>
<td>The caller-supplied ECB is inaccessible.</td>
</tr>
<tr>
<td>2D</td>
<td>The caller’s ASID list is inaccessible.</td>
</tr>
</tbody>
</table>
Table 26  Reason codes for RC=8 on message BMC110980E (part 2 of 2)

<table>
<thead>
<tr>
<th>Reason code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2E</td>
<td>The caller’s SUMLIST/SUMLSTA is inaccessible.</td>
</tr>
<tr>
<td>2F</td>
<td>The caller’s SUBPLST is inaccessible.</td>
</tr>
<tr>
<td>30</td>
<td>The caller’s KEYLIST is inaccessible.</td>
</tr>
<tr>
<td>31</td>
<td>Copies of the SLIP register and PSW are inaccessible.</td>
</tr>
<tr>
<td>32</td>
<td>The caller-supplied SRB is inaccessible.</td>
</tr>
<tr>
<td>33</td>
<td>The version number in the parameter list is not valid.</td>
</tr>
<tr>
<td>34</td>
<td>The caller’s LISTD is inaccessible.</td>
</tr>
<tr>
<td>35</td>
<td>The caller’s SUMLISTL is inaccessible.</td>
</tr>
<tr>
<td>36</td>
<td>The parameter list contains conflicting parameters.</td>
</tr>
<tr>
<td>37</td>
<td>The ID is longer than 50 characters.</td>
</tr>
<tr>
<td>38</td>
<td>The ID is not addressable.</td>
</tr>
<tr>
<td>39</td>
<td>The PSWREGS area is an incorrect length.</td>
</tr>
<tr>
<td>3A</td>
<td>The PSWREGS area is not addressable.</td>
</tr>
<tr>
<td>3B</td>
<td>The symptom record is not valid.</td>
</tr>
<tr>
<td>3C</td>
<td>The symptom record is not addressable.</td>
</tr>
<tr>
<td>3D</td>
<td>The DEB for the caller-supplied DCB is inaccessible.</td>
</tr>
<tr>
<td>3E</td>
<td>SVC dump is already using the maximum amount of virtual storage (as determined by the installation, using the MAXSPACE parameter on the CHNGDUMP command) to process other dumps.</td>
</tr>
<tr>
<td>3F</td>
<td>The caller-supplied STRLIST area is inaccessible.</td>
</tr>
<tr>
<td>40</td>
<td>The caller-supplied INTOKEN area is inaccessible.</td>
</tr>
<tr>
<td>41</td>
<td>The caller-supplied REMOTE area is inaccessible.</td>
</tr>
<tr>
<td>42</td>
<td>The caller-supplied PROBDESC area is inaccessible.</td>
</tr>
<tr>
<td>43</td>
<td>The caller-supplied JOBLIST area is inaccessible.</td>
</tr>
<tr>
<td>44</td>
<td>The caller-supplied DSPLIST area is inaccessible.</td>
</tr>
<tr>
<td>45</td>
<td>The caller-supplied REMOTE area is not valid. The length of a field in the REMOTE area is specified as less than 4 bytes.</td>
</tr>
<tr>
<td>46</td>
<td>SVC dump stopped the dump because the system resources manager (SRM) detected a critical shortage of auxiliary storage.</td>
</tr>
</tbody>
</table>

Define the general problem

By answering the following questions, you will begin to refine or isolate the problem that you are having:

- What PFX product messages or IMS messages were issued before and after the problem occurred (job log, control region, dependent region, and CICS region)?
Define PFX registration problems

- Are any system completion codes (abends) issued that explain why the system abnormally terminated PFX?

- BMC Customer Support will ask for the circumstances that caused the problem. Circumstances that might be useful in diagnosing the problem include the following information:
  - starting and stopping a message region
  - dynamically creating and collapsing CICS threads
  - peak system stress
  - other system component failures

- Has software maintenance has been applied to any software components?

- Are other BMC products installed (DELTA/IMS, APPLICATION RESTART CONTROL, and/or BATCH CONTROL FACILITY)? These products share a common DFSRRC00 module.

- Does the problem affect one or more of the following components:
  - particular system
  - DBD
  - PSB/transaction
  - PFX index (Is all PFX index processing affected?)

- How often does the problem occur? Is the problem intermittent or persistent?

- If the problem is in another vendor’s product and PFX might be involved, are any other software vendors working on the problem? If so, what is the ETR number or who is the other vendor contact?

Define PFX registration problems

The following items pertain to proper PFX registration:

- Review the primary DBD. Has the index been properly defined?

- Review the PSB source. Has the PCB PROCOPT or PROCSEQ been properly defined?

- Review the output in the PFXPRINT DD. Were any error messages issued by PFXAGEN?
Define index retrieval or maintenance problems

The following questions pertain to PFX index retrieval and maintenance issues:

- Is the PSB or DBD defined to IMS?
- Was dynamic allocation defined for the secondary index database?
- Have you performed index validation by using the VERIFY command?
  
  If the index is out of sync with the primary DEDB, has PFX been deactivated at any time, or have the source and secondary index databases been recovered?
- Is a sparse or partitioning exit used to maintain the index?
- Does the program linkage section match the actual PSB?
- What is the DL/I SSA call sequence that causes the problem?
- What is the DL/I status code or key feedback?
- Can the problem be recreated successfully by using DFSDDLT0?

Provide duplication and resolution information

BMC Customer Support might ask you to send documentation to reproduce a problem. You might be asked to provide the following items:

- job logs and SYSOUT
- SVC dump (if automatically generated by product)
- DLT0 scripts that recreate the problem
- DBDLIB and DBD source members
- PSBLIB and PSB source members
Problem summary and documentation form

- ACBLIB members
- PFXLIB members
- PFXDUMP DD output (SVC dump generated by PFX Dump utility discussed in “PFX Dump tool” on page 204)
- PFXTRACE DD output (generated by PFX Trace utility discussed in “PFX Trace tool” on page 204)
- sparse index exit, partition exit, and randomizer exit source
- monitor reports and job history statistics (CPU, Elapsed Time, EXCPs)

If a problem occurs, have the symptom dump available when you call BMC Customer Support. Save the SVC dump until the problem has been resolved to your satisfaction. BMC Customer Support will generally ask for the PSW, PSW data, and general purpose registers at the time of the abend.

BMC Customer Support will determine whether it is necessary to send a dump. If BMC Customer Support requests that you supply a dump, provide the appropriate accompanying documentation. Use the BMC web site to submit the dump via FTP. Contact BMC Customer Support for instructions.

**NOTE**
Abbreviated dumps do not provide the type of information that is required for problem resolution. Please ensure that a complete dump is taken.

Problem summary and documentation form

The problem summary and documentation form in Table 27 should be copied, filled out, and faxed to BMC Customer Support for their understanding and resolution of your problem.
### Table 27  Problem summary and documentation form (part 1 of 4)

**Fast Path Indexer/EP**

**PROBLEM SUMMARY**

<table>
<thead>
<tr>
<th>Issue Number:</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company:</td>
<td></td>
</tr>
<tr>
<td>Contact:</td>
<td></td>
</tr>
<tr>
<td>Phone:</td>
<td></td>
</tr>
<tr>
<td>Email:</td>
<td></td>
</tr>
</tbody>
</table>

---------- **Software Product Levels**  ----------

- **PFX Version.Release.Maintenance Level:** _______.____.____
- **IMS Version.Release.PUT Level:** _______.____.____
- **CICS Version.Release.PUT Level:** _______.____.____
- **MVS Version.Release:** _______.____.____

---------- **Problem Description**  ----------

**PFX Product Messages:**
- **BMC**
- **BMC**
- **BMC**

**Other Messages (IMS, MVS, or JES):**
- __________________
- __________________

**System Completion Code= _____  Reason Code= ______**

**Circumstances that caused the problem:**
- __________________
- __________________
- __________________

**Problem Frequency:  __ Intermittent  __ Continuous**

**Has maintenance been applied to any software?  __ Y or N**

**Describe:**
- __________________
- __________________
- __________________
<table>
<thead>
<tr>
<th>Product</th>
<th>Installed</th>
<th>Y or N</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELTA/IMS</td>
<td></td>
<td>__ __</td>
</tr>
<tr>
<td>APPLICATION RESTART CONTROL (AR/CTL)</td>
<td></td>
<td>__ __</td>
</tr>
<tr>
<td>BATCH CONTROL FACILITY (BCF)</td>
<td></td>
<td>__ __</td>
</tr>
</tbody>
</table>

### PFX Registration Problem Information

**DBD Source:**

- **PFXCHILD NAME=** ________________________________
  - ACCESS=DEDB or
  - ACCESS=SHISAM or HISAM

- **PFXXDFLD NAME=** ________________________________
  - SEGMENT=________.
  - SRCH=______________________________.
  - DDATA=______________________________.
  - NULLVAL=__________.
  - EXTRTN=__________.
  - PARTEXIT=__________.

**PSB Source:**

Alternate Processing Sequence:

- **PCB TYPE=DB,**
  - PROCSEQ=__________.
  - DBDNAME=__________.
  - KEYLEN=______.
  - PROCOPT=___

Stand-Alone Index Processing:

- **PCB TYPE=DB,**
  - DBDNAME=__________.
  - KEYLEN=__________.
  - PROCOPT=G

**SENSEG...**

PFXAGEN Error Messages (PFXPRINT DD):

BMC_________ _________________________________________________
BMC_________ _________________________________________________
BMC_________ _________________________________________________
BMC_________ _________________________________________________

---

**Table 27**  Problem summary and documentation form (part 2 of 4)
Table 27  Problem summary and documentation form (part 3 of 4)

ACBGEN Messages Issued During PFXAGEN (SYSPRINT DD):


Staging, active, inactive ACBLIB/PFXLIB coordinated?    Y or N

--------PFX Index Retrieval Or Maintenance Information--------

Y or N

Is the PSB or DBD defined to IMS?                        __  __
Is dynamic allocation properly defined to allocate the index data base to IMS?  __  __
Has index validation been performed by using the VERIFY command?  __  __
Is a sparse or partitioning exit used to maintain the index?  __  __
Does the program linkage section match the actual PSB?  __  __
Can the problem be recreated by using DFSDDLT0?  __  __

DLI Status Code: ___ Key Feedback Area:________________________

---------------------Problem Documentation----------------------
The following checklist identifies documentation that might be requested for problem resolution. Check the documentation that has been collected.

Job logs/SYSOUT ______
SVC Dump ______
DLTO Scripts ______
DBDLIB and DBD Source ______
PSBLIB and PSB Source ______
ACBLIB Members ______
PFXLIB Members ______
PFXDUMP DD Output ______
PFXTRACE DD Output and DL/I Trace Output ______
Sparse Index, Partition, and Randomizer Exit Source ______
Monitor Reports/Job History Statistics ______
### Abend Summary

IEA995I SYMPTOM DUMP OUTPUT

SYSTEM COMPLETION CODE= ____________  REASON CODE= ____________

PSW AT TIME OF ERROR _______   _________  ILC ____  INTC _____

ACTIVE LOAD MODULE= _____  ADDRESS= ____  OFFSET= __________

DATA AT PSW_____________- __________  _________  _________

GPR 0-3    _____________  __________  _________  _________

GPR 4-7    _____________  __________  _________  _________

GPR 8-11   _____________  __________  _________  _________

GPR 12-15  _____________  __________  _________  _________

END OF SYMPTOM DUMP

---

Table 27  Problem summary and documentation form (part 4 of 4)
The Fast Path Indexer sample application (PFX-SAMP) is provided with the Fast Path Indexer/EP product for your optional use. This document discusses the following topics:

Overview ................................................................. 216
  Distribution data sets ........................................... 216
  PFX-SAMP components .......................................... 216
Setting up PFX-SAMP .................................................. 217
  Defining PFX-SAMP to IMS ................................... 217
  PFX-SAMP JCL ...................................................... 218
PFX-SAMP databases .................................................. 220
  Primary DEDB .................................................... 220
  Secondary index databases .................................... 222
  Journal database ................................................ 224
Using PFX-SAMP online transactions and panels ................... 224
  Online transactions .............................................. 225
  Primary DEDB transaction instructions ...................... 226
  Menu and panel descriptions ................................ 228
  Online practice session ....................................... 244
Overview

The Fast Path Indexer sample application (PFX-SAMP) is provided with the Fast Path Indexer/EP product for your optional use. You can use PFX-SAMP to perform the following tasks related to Fast Path Indexer/EP:

- demonstrate the product features and functions of Fast Path Indexer/EP
- provide an application for immediate product testing
- verify that the product has been installed correctly on your system by executing certain transactions within PFX-SAMP

PFX-SAMP can also be used to help you gain familiarity with the IMS Fast Path primary DEDBs, regardless of whether or not you have installed the Fast Path Indexer/EP product.

NOTE

The Fast Path Indexer sample application is designed to create and process PFX indexes; it cannot create or process IBM native indexes.

Distribution data sets

The FPU product distribution tape, which includes the Fast Path Indexer/EP product, contains four data sets that are specific to PFX-SAMP:

- IMS source data set – contains source to create the DFSMDAs, DBDs, PSBs, and MFS
- program source data set – contains code for the COBOL programs that drive the transactions available in the online practice session
- REPRO primary DEDB – the IDCAMS REPRO data set used to create the primary DEDB
- REPRO journal database – the IDCAMS REPRO data set used to create the journal database

PFX-SAMP components

PFX-SAMP consists of the following components:
Setting up PFX-SAMP

You must generate all resources such as PSBs, ACBs, and online panels for PFX-SAMP. However, these resources will not be used until PFX-SAMP is used or Fast Path Indexer/EP is installed.

NOTE

PFX-SAMP must be installed only on IMS Transaction Manager (IMS/TM). The CICS transaction management subsystem is not supported.

Defining PFX-SAMP to IMS

You must provide your systems programmer with the PFXSTG1 member found in the IVPSRC data set. The PFXSTG1 member has the application and database specifications that must be included in your IMS definition.

To ensure that the IMS control region can allocate the PFX-SAMP databases, you must select one of the following methods:

- create DFSMDA members
- include DD statements in the control region JCL
- register the databases with DBRC
PFX-SAMP JCL

The Fast Path/EP sample library (PFPSAMP or IMSAMP) contains sample JCL for PFX-SAMP.

Tailor all of the following jobs by changing the following statements:

- JOB statement
- JCL statements as indicated by comments in each job

The JCL should conform to your installation standards.

To tailor the JCL

1 Tailor and execute PFXJOB01 in the Fast Path/EP sample library.
   This job is a DBDGEN for the primary (DIVSSD) and journal (DIVSJR) DEDBs.

   **WARNING**
   Do not change any of the DBD parameters. If you do, the DBDGEN will not match the distributed databases.

2 Tailor and execute PFXJOB02 in the Fast Path/EP sample library.
   This job is a PSBGEN for a dummy PSB (DIVSPDMY). This is required for the ACBGEN in Step 3.

3 Tailor and execute PFXJOB03 in the Fast Path/EP sample library.
   This job is an ACBGEN that builds a DMB for DIVSSD and DIVSJR in ACBLIB. If Fast Path Indexer/EP is installed, Step 6 rebuilds the ACBs using the PFXAGEN utility included with Fast Path Indexer/EP.

4 Tailor and execute PFXJOB04 in the Fast Path/EP sample library.
   This job is a PSBGEN for PSBs DIVSP0 through DIVSP5. These PSBs are required for the ACBGEN in Step 5.

5 Tailor and execute PFXJOB05 in the Fast Path/EP sample library.
   This job is an ACBGEN that builds ACBs used by the online transactions DIVST0 through DIVST5. When executed correctly, this step ends with a return code of 04 or less.

6 Tailor and execute PFXJOB06 in the Fast Path/EP sample library.
This job is the index definition process. It executes the necessary DBDGENs and PSBGENs for the indexes to be associated with the primary DEDB. It also executes PFXAGEN, which builds the appropriate ACB and PFX control blocks.

7 Tailor and execute PFXJOB07 in the Fast Path/EP sample library.

This job allocates the VSAM cluster for the four secondary index databases that are created for PFX-SAMP. Change each occurrence of parameter VOLUME(??????) to an appropriate volume name.

8 Tailor and execute PFXJOB08 in the Fast Path/EP sample library.

This job allocates the VSAM cluster for the primary and journal DEDBs, and uses IDCAMS REPRO to populate these databases with data. Change each occurrence of parameter VOLUME(??????) to an appropriate volume name.

9 Tailor and execute PFXJOB09 in the Fast Path/EP sample library.

Use the BUILD command with the IX subcommand to create and load the secondary index databases. For more information about using BUILD and IX, see Chapter 9, “Index build utility.”

10 Tailor and execute PFXJOB10 in the Fast Path/EP sample library.

Use MFSGEN to generate the online panels to your TFORMAT library.


Tailor the JCL by changing the statements marked CHANGE. Compile the COBOL programs used by the online transactions.

12 Allocate PFX-SAMP databases to IMS.

You might use one of three optional methods:

- Code JCL statements in the IMS control region and the DLI/SAS region.
- Generate dynamic allocation members for PFX-SAMP databases. See member ‘PFXDBALC’ in the IVPSRC data set for examples.
- Register PFX-SAMP databases with DBRC.

13 Refresh the ACB and PFX online control blocks in your IMS online system.

Because ACBLIB and PFXLIB have been modified, you must ensure that the online IMS used for PFX-SAMP is using the correct (active) ACBLIBx and PFXLIBx libraries.
For an example that shows how to update PFXLIBx with new PFX members, see member PFXOLC in the Fast Path/EP sample library.

PFX-SAMP databases

This section describes the PFX-SAMP databases. Three types of databases interact to represent sample course descriptions for a fictitious university information system:

- the primary DEDB that contains information about university courses, instructors, and students
- four secondary index databases, each of which provides a unique alternate method for accessing course information
- the journal database that is used to record update activity to the primary and index DEDBs

The primary and journal DEDBs are used during the following activities:

- an online practice session with PFX-SAMP to become familiar with Fast Path Indexer/EP
- verifying correct installation of Fast Path Indexer/EP on your system

Primary DEDB

The sample primary DEDB is simple but should be sufficient for online practice sessions and for verifying correct installation of Fast Path Indexer/EP.

Figure 98 shows DFSMDA macros and the embedded DBD names for the PFX-SAMP databases. If you do not want to use dynamic allocation, put the following data set names in the IMS startup procedure. An example of the dynamic allocation control cards can be found in member PFXDBALC in the IVPSRC data set.

Figure 98  Sample Fast Path databases for IMSID=IMS5 (part 1 of 2)

```plaintext
*-----------------------------------------------------------------------------------*
* PFX SAMPLE APPLICATION FAST PATH DATABASES FOR IMSID=IMS5 *
*-----------------------------------------------------------------------------------*
DFSMDA TYPE=FPDEDB, DBNAME=DIVSSD
DFSMDA TYPE=DATASET, DDNAME=DIVSSD1, DSNAME=BMC.PFP.DIVSSD1, DISP=SHR
DFSMDA TYPE=FPDEDB, DBNAME=DIVSJR
```
Figure 98 Sample Fast Path databases for IMSID=IMS5 (part 2 of 2)

| DFSMDA TYPE=DATASET,DDNAME=DIVSJR1,DSNAME=BMC.PFP.DIVSJR1, DISP=SHR |
| DFSMDA TYPE=DATABASE,DBNAME=DIVSI1 |
| DFSMDA TYPE=DATASET,DDNAME=DIVSI11,DSNAME=BMC.PFP.DIVSI11, DISP=SHR |
| DFSMDA TYPE=DATABASE,DBNAME=DIVSI2 |
| DFSMDA TYPE=DATASET,DDNAME=DIVSI121, DSNAME=BMC.PFP.DIVSI121, DISP=SHR |
| DFSMDA TYPE=DATABASE,DBNAME=DIVSI3 |
| DFSMDA TYPE=DATASET,DDNAME=DIVSI31,DSNAME=BMC.PFP.DIVSI31, DISP=SHR |
| DFSMDA TYPE=DATABASE,DBNAME=DIVSI4 |
| DFSMDA TYPE=DATASET,DDNAME=DIVSI41,DSNAME=BMC.PFP.DIVSI41, DISP=SHR |

Figure 99 illustrates the structure of the primary DEDB (DIVSSD).

Figure 99 Example primary DEDB structure

- The root segment (course) is keyed by course number and contains the course name.
- The location segment is keyed by location code and contains the location name.
Each location can have multiple instructor segments keyed by instructor number and multiple student segments keyed by student number.

The prerequisite segment is keyed by prerequisite course number and contains the prerequisite course name.

Secondary index databases

Four secondary index databases are associated with the primary DEDB: two are in HISAM format, and two are in SHISAM format.

PFX-SAMP uses an index on the location code to list all courses that are taught at a specific location.

The Location Code index shown in Figure 100 is a HISAM-format compact index that contains the location code as the key. To force unique keys, the course number from the concatenated key is used as a subsequence field.

Figure 100 Location code index (DIVSI1)

<table>
<thead>
<tr>
<th>Location Code</th>
<th>Subsequence Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Number</td>
<td>Concatenated Key of Target</td>
</tr>
</tbody>
</table>

PFX-SAMP also uses a SHISAM-format index on instructor social security number (SSN), which allows access and update capabilities to the Instructor segment using the instructor’s SSN. This is a unique index because all SSNs are unique; the instructor name and phone number are carried as duplicate data.

Sparse indexing is used with the Instructor SSN index (Figure 101) to avoid creating index entries for SSNs containing all zeros.
To accommodate students who forget which courses they are taking, their course locations, or their student numbers, another HISAM-format index accesses the application using the student surname. To force unique keys, the student number from the source segment and the location code and course number from the concatenated key of the source segment are used as subsequence fields. Because the Surname Index also carries the surname and first name as duplicate data, the index can be processed as a stand-alone database, independent of the primary DEDB (Figure 102).

A fourth index on the area code of the student’s phone number is structured in SHISAM format. To force unique keys, the student number from the source segment and the location code and course number from the concatenated key of the source segment are used as subsequence fields. The student’s phone number, surname, and first name are carried as duplicate data (Figure 103).
The journal database is used by the PFX-SAMP to record update activity to the primary DEDB. Input transactions DIVST1 through DIVST5 write sequential dependents to the journal database. The transaction code is the key of the root segment.

If the journal database becomes full, you should delete and reallocate it, then initialize it using DBFUMINO. If a transaction attempts to journal a transaction and the appropriate root segment is missing, the transaction inserts the root segment.

Using PFX-SAMP online transactions and panels

This section explains how to use the PFX-SAMP online IMS transactions and panels. You can use these transactions and panels to complete the following:

- an online practice session to become familiar with IMS Fast Path and DEDB databases (see page 244)
- installation verification for Fast Path Indexer/EP (by successfully executing certain transactions)
Online transactions

PFX-SAMP includes 11 online IMS transactions which are summarized in Table 28. The corresponding PF keys are pre-defined to execute these transactions. The PF key assignments are also listed on the PRIMARY MENU shown in Figure 104. For example, press PF12 to return to the PRIMARY MENU from any other transaction or press PF16 to access the STUDENT MENU from any other transaction.

The source code for the COBOL programs that drive these transactions is located in IVPSRC.

**WARNING**

Do not make any changes to these COBOL programs.

Panel examples shown in this section (DIVST0 through DIVST9 and DIVSTA) are IMS panels.

**NOTE**

Transactions DIVST6 through DIVST9 and DIVSTA are available only if you have already installed Fast Path Indexer/EP.

<table>
<thead>
<tr>
<th>IMS transaction codes</th>
<th>PF key</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVST0</td>
<td>PF12</td>
<td>Display PRIMARY MENU.</td>
</tr>
<tr>
<td>DIVST1</td>
<td>PF13</td>
<td>Add, change, delete, or browse Course segment of the primary DEDB.</td>
</tr>
<tr>
<td>DIVST2</td>
<td>PF14</td>
<td>Add, change, delete, or browse Location segment of the primary DEDB.</td>
</tr>
<tr>
<td>DIVST3</td>
<td>PF15</td>
<td>Add, change, delete, or browse Instructor segment of the primary DEDB.</td>
</tr>
<tr>
<td>DIVST4</td>
<td>PF16</td>
<td>Add, change, delete, or browse Student segment of the primary DEDB.</td>
</tr>
<tr>
<td>DIVST5</td>
<td>PF17</td>
<td>Add, change, delete, or browse Prerequisite segment of the primary DEDB.</td>
</tr>
<tr>
<td>DIVST6</td>
<td>PF18</td>
<td>Access the Student Surname secondary index database to obtain a student name.</td>
</tr>
<tr>
<td>DIVST7</td>
<td>PF19</td>
<td>Access the Location Code secondary index database to list courses for a specified location.</td>
</tr>
<tr>
<td>DIVST8</td>
<td>PF20</td>
<td>Change, delete, or browse instructor data in the primary DEDB via the Instructor SSN secondary index database</td>
</tr>
<tr>
<td>DIVST9</td>
<td>PF21</td>
<td>Access the Student secondary index database via student surname.</td>
</tr>
<tr>
<td>DIVSTA</td>
<td>PF22</td>
<td>Access the Student Area Code secondary index database via area code.</td>
</tr>
</tbody>
</table>
Primary DEDB transaction instructions

This section provides a brief overview of how to add, change, delete, browse, and get segments. Refer to the panel descriptions that follow this section for detailed instructions.

Transactions DIVST1, DIVST2, DIVST3, DIVST4, DIVST5, and DIVST8 allow you to add, change, delete, and browse segments from the primary DEDB. Get (G) and Get Next within Parent (GNP) processing allows you to view existing segments in the database.

Add (A)

Follow these instructions to add a segment:

1. Type A in the SELECT FUNCTION field.
2. Enter values for the required fields.
3. Press Enter to accept the transaction.
4. Review the message in the lower left of the panel to ensure that the addition is accepted. If the transaction is not accepted, review the values entered for the fields and correct the information.

Change (C)

Follow these instructions to change a segment:

1. Use the Browse (B) or Get-type (G/GNP) function to display the segment you want to change.
2. Type C in the SELECT FUNCTION field.
3. Enter new values for the fields that can be changed.
4. Press Enter to accept the transaction.
5. Review the message in the lower left of the panel to ensure that the change is accepted. If the transaction is not accepted, review the values entered for the fields and correct the information.

Delete (D)

Follow these instructions to delete a segment:
1 Use the Browse (B) or Get-type (G/GNP) function to display the segment you want to delete.

2 Type D in the SELECT FUNCTION field.

3 Press Enter to accept the transaction.

4 Review the message in the lower left of the panel to ensure that the deletion is accepted. If the transaction is not accepted, review the values entered and correct the information.

**Browse (B)**

Follow these instructions to browse a segment:

1 Type B in the SELECT FUNCTION field.

2 Enter values for the required fields.

3 Press Enter to accept the transaction.

4 Review the message in the lower left of the panel to ensure that the requested segment is displayed. If the requested segment is not displayed, review the values entered for the fields and correct the information.

**Get (G) or Get Next within Parent (GNP)**

Follow these instructions to get a segment or to get the next segment within a parent:

1 Type G in the SELECT FUNCTION field.

2 Enter values for the required fields.

3 Press Enter to accept the transaction.

4 Review the message in the lower left of the panel to ensure that the requested segment is displayed. If the requested segment is not displayed, review the values entered for the fields and correct the information.

5 Continue to press Enter to view the remaining segments until the message “NO MORE segmentType ON FILE” is displayed.
Menu and panel descriptions

This section presents detailed instructions for completing each menu and panel, including sample panels and instructions for completing the data entry fields. Refer to this section as needed when completing the online practice session beginning on page 244.

**PRIMARY MENU (DIVST0)**

Options 1 through 9 and A on the PRIMARY MENU (Figure 104) allow you to select a transaction for database processing (transaction DIVST0):

- Options 1 through 5 allow you to maintain the primary DEDB.
- Options 6 through 8 allow you to access the primary DEDB via one of its associated secondary index databases.
- Options 9 and A allow you to access a secondary index database for stand-alone processing.

*Figure 104 PRIMARY MENU*

TRAN: DIVST0  PFX SAMPLE APPLICATION SYSTEM  DATE: :::::::::::
LTRM: ::::::::  (PRIMARY MENU)  TIME: ::::::::
MOD : PFXMOD00

SELECT OPTION .. _  1. COURSE  (PF13)
   2. LOCATION  (PF14)
   3. INSTRUCTOR  (PF15)
   4. STUDENT  (PF16)
   5. PREREQUISITE  (PF17)
   
   ACCESS VIA INDEX:
   ***USE ONLY IF FAST PATH INDEXER/EP IS INSTALLED***
   
   6. STUDENT NAME  (PF18)
   7. LOCATION CODE  (PF19)
   8. INSTRUCTOR SOCIAL SECURITY NUMBER  (PF20)
   9. STUDENT NAME (INDEX ONLY)  (PF21)
   A. STUDENT AREA CODE (INDEX ONLY)  (PF22)

:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
ENTER=OPTION  PF12=CANCEL  PF13 THRU PF22 = OPTIONS 1 THRU A.
SELECT OPTION

Type the number associated with the transaction you want to use and press Enter. You can also press the corresponding PF key to access the transaction. Then the selected transaction panel appears.

COURSE MENU (DIVST1)

The COURSE MENU (Figure 105) allows you to add, change, delete, browse, and obtain the Course segments on the university information database (transaction DIVST1).

For generic information about how to add, change, delete, browse, and get segments, see “Primary DEDB transaction instructions” on page 226.

Figure 105  COURSE MENU

SELECT FUNCTION (. . . .) (A=ADD, C=CHANGE, D=DELETE, B=BROWSE, G=GN)

COURSE NUMBER . . . . _____

COURSE NAME . . . . ______________________________

SELECT FUNCTION

Valid entries include the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>You must enter a course number and course name.</td>
</tr>
<tr>
<td>C</td>
<td>You can change the course name only.</td>
</tr>
<tr>
<td>D</td>
<td>You must enter an existing course number. You can also use the B or G functions to display a segment, then type D to delete the segment.</td>
</tr>
</tbody>
</table>
### COURSE NUMBER

Type the course number (five digits). Blanks are zero-filled. This is the key field for the course segment.

### COURSE NAME

Type the course name (up to 30 alphanumeric characters).

---

**NOTE**

Use the **ERASE EOF** key to blank a field. Do not use spaces.

---

### LOCATION MENU (DIVST2)

The LOCATION MENU (Figure 106) allows you to add, change, delete, browse, and obtain Location segments related to a selected course (transaction DIVST2).

For generic information about how to add, change, delete, browse, and get segments, see “Primary DEDB transaction instructions” on page 226.
Figure 106  LOCATION MENU

SELECT FUNCTION . . . _ (A=ADD,C=CHANGE,D=DELETE,B=BROWSE,G=GNP)

COURSE NUMBER . . . .

LOCATION CODE . . . .

LOCATION NAME . . . .

SELECT FUNCTION

Valid entries include the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>You must enter new course number, location code, and location name.</td>
</tr>
<tr>
<td>C</td>
<td>You can change the location name only.</td>
</tr>
<tr>
<td>D</td>
<td>You must enter an existing course number and location code. You can also use the B or G functions to display a location segment and then type D to delete the segment.</td>
</tr>
<tr>
<td>B</td>
<td>You must enter an existing course number and location code.</td>
</tr>
<tr>
<td>G</td>
<td>You must enter an existing course number. Leave location code blank to view the first location segment in the database. You can also type G after an add, change, delete, or browse to view the next location segment for the selected course.</td>
</tr>
</tbody>
</table>

COURSE NUMBER

Type a course number (five digits). This is part of the segment key.

LOCATION CODE

Type a location code (up to four alphabetic characters). This is part of the segment key.
LOCATION NAME

Type a location name (up to 30 alphanumeric characters).

**NOTE**

Use the ERASE EOF key to blank a field. Do not use spaces.

INSTRUCTOR MENU (DIVST3)

The INSTRUCTOR MENU (Figure 107) allows you to add, change, delete, browse, and obtain instructor segments that are related to a selected course and location (transaction DIVST3).

For generic information about how to add, change, delete, browse, and get segments, see “Primary DEDB transaction instructions” on page 226.

**Figure 107  INSTRUCTOR MENU**

```
TRAN: DIVST3   PFX SAMPLE APPLICATION SYSTEM   DATE:   :::::::::
LTRM:   ::::::::   (INSTRUCTOR MENU)   TIME:   :::::::::
MOD : PFXM003

SELECT FUNCTION . . . .  (A=ADD,C=CHANGE,D=DELETE,B=BROWSE,G=GNP)

COURSE NUMBER . . . . _____
LOCATION CODE . . . . __
INSTRUCTOR ID . . . . _____
INSTRUCTOR NAME . . ______________________________
SOC SEC # . . . . . . . . . . __
PHONE NUMBER . . . . ___ - ___ - ____

ENTER=FUNCTION  PF12=CANCEL  PF13 THRU PF22 = OPTIONS 1 THRU A.
```

SELECT FUNCTION

Valid entries include the following options:
COURSE NUMBER

Type a course number (five digits). This is part of the segment key.

LOCATION CODE

Type a location code (up to four alphabetic characters). This is part of the segment key.

INSTRUCTOR ID

Type an instructor ID (five digits). This is part of the segment key.

INSTRUCTOR NAME

Type an instructor name (up to 30 alphanumeric characters).

SOC SEC #

Type a Social Security number (nine digits). Do not include dashes.

PHONE NUMBER

Type a phone number (10 digits). Include the area code.

NOTE

Use the ERASE EOF key to blank a field. Do not use spaces.
STUDENT MENU (DIVST4)

The STUDENT MENU (Figure 108) allows you to add, change, delete, browse, and obtain student segments that are related to a selected course and location (transaction DIVST4).

For generic information about how to add, change, delete, browse, and get segments, see “Primary DEDB transaction instructions” on page 226.

Figure 108 STUDENT MENU

| MOD : PFXM004 | |

SELECT FUNCTION . . . _ (A=ADD, C=CHANGE, D=DELETE, B=BROWSE, G=GNP)

COURSE NUMBER . . . .
LOCATION CODE . . . .
STUDENT NUMBER . . . .
SURNAME . . . . . . ______________________________
FIRST NAME. . . . ______________________________
PHONE NUMBER. . . . ___ - ___ - ____ SOC SEC #. . . . ______
ADDRESS1 . . . . . . __________________________________
ADDRESS2 . . . . . . __________________________________
ADDRESS3 . . . . . . __________________________________
ADDRESS4 . . . . . . __________________________________

ENTER=FUNCTION PF12=CANCEL PF13 THRU PF22 = OPTIONS 1 THRU A.

SELECT FUNCTION

Valid entries include the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>You must enter an existing course number and location code and a new student number, surname, and first name. Phone number, soc sec #, and address1 through address4 are optional.</td>
</tr>
<tr>
<td>C</td>
<td>You can change the surname, first name, phone number, soc sec #, and address1 through address4.</td>
</tr>
<tr>
<td>D</td>
<td>You must enter an existing course number, location code, and student number. You can also use the B or G functions to display a student segment and then type D to delete the segment.</td>
</tr>
<tr>
<td>Option</td>
<td>Use</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
</tr>
<tr>
<td>B</td>
<td>You must enter an existing course number, location code, and student number.</td>
</tr>
<tr>
<td>G</td>
<td>You must enter an existing course number and location code. Leave student number blank to view the first student segment in the database for the selected course and location. You can also type G after an add, change, delete, or browse to view the next segment for the selected course and location.</td>
</tr>
</tbody>
</table>

**COURSE NUMBER**

Type a course number (five digits). This is part of the segment key.

**LOCATION CODE**

Type a location code (up to four alphabetic characters). This is part of the segment key.

**STUDENT NUMBER**

Type a student number (six digits). This is part of the segment key.

**SURNAME**

Type a surname (up to 25 alphanumeric characters).

**FIRST NAME**

Type a first name (up to 20 alphanumeric characters).

**PHONE NUMBER**

Type a phone number (10 digits). Include the area code.

**SOC SEC #**

Type a Social Security number (nine digits). Do not include dashes.

**ADDRESS1, 2, 3, 4**

Type an address (up to 35 alphanumeric characters for each line).

---

**NOTE**

Use the ERASE EOF key to blank a field. Do not use spaces.
PREREQUISITE MENU (DIVST5)

The PREREQUISITE MENU (Figure 109) allows you to add, change, delete, browse, and obtain prerequisite segments for a selected course (transaction DIVST5).

For generic information about how to add, change, delete, browse, and get segments, see “Primary DEDB transaction instructions” on page 226.

Figure 109  PREREQUISITE MENU

| TRAN: DIVST5 | PFX SAMPLE APPLICATION SYSTEM | DATE: ::::: |
| LTRM: :::::::: | (PREREQUISITE MENU) | TIME: :::::::: |
| MOD : PFXMDS05 |

SELECT FUNCTION . . . . (A=ADD,C=CHANGE,D=DELETE,B=BROWSE,G=GNP)

COURSE NUMBER . . . . __
PREREQ NUMBER . . . . __
PREREQ NAME . . . . ______________________________

ENTER=FUNCTION PF12=CANCEL PF13 THRU PF22 = OPTIONS 1 THRU A.

SELECT FUNCTION

Valid entries include the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>You must enter an existing course number and a new prereq number and prereq name.</td>
</tr>
<tr>
<td>C</td>
<td>You can change the prereq name only.</td>
</tr>
<tr>
<td>D</td>
<td>You must enter an existing course number and prereq number. You can also use the B or G functions to display a prerequisite segment and then type D to delete the segment.</td>
</tr>
</tbody>
</table>
### COURSE NUMBER

Type a course number (five digits). This is part of the segment key.

### PREREQ NUMBER

Type a prerequisite number (five alphabetic characters). This is part of the segment key.

### PREREQ NAME

Type a prerequisite name (up to 30 alphanumeric characters).

---

**NOTE**

Use the ERASE EOF key to blank a field. Do not use spaces.

---

### ACCESS VIA STUDENT NAME panel (DIVST6)

The ACCESS VIA STUDENT NAME panel (Figure 110) allows you to access the student segments of a primary DEDB via an index on the student name (transaction DIVST6).

Type the student’s last name in the STUDENT NAME field and press Enter. A listing of all students with the same name spellings displays. Continue to press Enter until the message “END OF STUDENT LIST” displays in the lower left corner of the panel.

To select a particular student to view or update, type information in the COURSE, LOCATION, and STUDENT# fields. This information is in the listing at the bottom of the panel. Then press PF16 to access the STUDENT MENU for the selected student.

---

**NOTE**

This transaction is available only if Fast Path Indexer/EP is installed on your system.
**Figure 110  ACCESS VIA STUDENT NAME panel**

![ACCESS VIA STUDENT NAME panel]

**STUDENT NAME**

Type a student last name (up to 25 alphabetic characters).

**COURSE**

Type a course number (five digits). This is part of the segment key.

**LOCATION**

Type a location code (up to four alphabetic characters). This is part of the segment key.

**STUDENT#**

Type a student number (up to six digits). This is part of the segment key.

**ACCESS VIA LOCATION CODE panel (DIVST7)**

The ACCESS VIA LOCATION CODE panel (Figure 111) allows you to access the location segments of the primary DEDB via an index on course location (transaction DIVST7).
Type the location code in the **LOCATION CODE** field and press **Enter**. A listing of all courses taught at the selected location displays. Continue to press **Enter** until the message “END OF COURSE LIST” displays in the lower left corner of the panel.

To select a particular course to view or update, type the course number in the **SELECT COURSE#** field and press **PF13**. The COURSE MENU for the selected course displays.

**NOTE**
This transaction is available only if Fast Path Indexer/EP is installed on your system.

**Figure 111   ACCESS VIA LOCATION CODE panel**

| TRAN: DIVST7 | DATE: 11/04/9 |
| LTRM: L3ACG2 | PFX SAMPLE APPLICATION SYSTEM |
| MOD: PFXMO07 | TIME: 14:55:5 |

LOCATION CODE . . . .
LOCATION NAME . . . :

SELECT COURSE#. . . (PRESS PF13 FOR COURSE) SSP:

COURSE# COURSE NAME

ENTER=MORE PF12=CANCEL PF13 THRU PF22 = OPTION 1 THRU A.

**LOCATION CODE**
Type a location code (four digits).

**SELECT COURSE#**
Type a course number (five digits). This is part of the segment key.
ACCESS VIA INSTRUCTOR SSN panel (DIVST8)

The ACCESS VIA INSTRUCTOR SSN panel (Figure 112) allows you to change, delete, browse, and obtain instructor segments by instructor social security number (transaction DIVST8). Since a SHISAM format is used for the index, instructor segments are displayed sequentially in numeric order by SSN.

For generic information about how to add, change, delete, browse, and get segments, see “Primary DEDB transaction instructions” on page 226.

NOTE

This transaction is available only if Fast Path Indexer/EP is installed on your system.

Figure 112  ACCESS VIA INSTRUCTOR SSN panel

SELECT FUNCTION . . . . (B=BROWSE, C=CHANGE, D=DELETE, G=GN)

INSTRUCTOR SSN. . . . _________

INSTRUCTOR ID . . . . ___

INSTRUCTOR NAME . . ______________________________

SOC SEC # . . . . . _________

PHONE NUMBER . . . . ___ - ___ - ____

ENTER=FUNCTION PF12=CANCEL PF13 THRU PF22 = OPTIONS 1 THRU A.

SELECT FUNCTION

Type one of the following selections:

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>You can change the instructor name, soc sec #, and phone number.</td>
</tr>
<tr>
<td>D</td>
<td>You must enter an existing instructor ssn. You can also use the B or G functions to display an instructor segment and then type D to delete the segment.</td>
</tr>
</tbody>
</table>
### INSTRUCTOR SSN

Type an instructor’s social security number (nine digits). Do not include dashes.

### INSTRUCTOR ID

Type an instructor ID (five digits).

### SOC SEC #

Type a social security number (nine digits). Do not include dashes.

### PHONE NUMBER

Type a phone number (10 digits). Include the area code.

---

**NOTE**

Use the `ERASE EOF` key to blank a field. Do not use spaces.

---

### STUDENT NAME INDEX ONLY panel (DIVST9)

The STUDENT NAME INDEX ONLY panel (Figure 113) is similar to the ACCESS VIA STUDENT NAME panel. Using STUDENT NAME INDEX ONLY, you can access the student segments by using the stand-alone database instead of the primary DEDB (transaction DIVST9). The secondary index database duplicates data from the primary DEDB.

Type the student’s last name in the **STUDENT NAME** field. When you press **Enter**, a listing of all students with exact or similar names displays. Continue to press **Enter** until the message “NO MORE STUDENTS” displays in the lower left corner.

To select a particular student to view or update, type information in the **COURSE**, **LOCATION**, and **STUDENT#** fields. This information is in the listing at the bottom of the panel. Then press **PF16** to access the STUDENT MENU for the selected student.
NOTE

This transaction is available only if Fast Path Indexer/EP is installed on your system.

Figure 113  STUDENT NAME INDEX ONLY panel

| STUDENT NAME | _________________ |
| SELECT COURSE | _____ |
| LOCATION | ____ |
| STUDENT# | ______ (PRESS PF16 FOR STUDENT NAME) |

STUDENT NAME

Type a student last name (up to 25 alphabetic characters).

COURSE

Type a course number (five digits). This is part of the segment key.

LOCATION

Type a location code (up to four alphabetic characters). This is part of the segment key.

STUDENT#

Type a student number (six digits). This is part of the segment key.
**STUDENT AREA CODE INDEX ONLY panel (DIVSTAA)**

The STUDENT AREA CODE INDEX ONLY panel (transaction DIVSTA) (see Figure 114) allows you to access the segments in the student area code index as a stand-alone database. This secondary index database duplicates data from the primary DEDB.

Type the student’s area code in the **STUDENT AREA CODE** field. When you press **Enter**, a listing of all students with the same area code displays. Continue to press **Enter** until the message “NO MORE STUDENTS” displays in the lower left corner.

To select a particular student to view or update, type information in the **COURSE**, **LOCATION**, and **STUDENT#** fields. This information is in the listing at the bottom of the panel. Then press **PF16** to access the **STUDENT MENU** for the selected student.

---

**NOTE**

This transaction is available only if Fast Path Indexer/EP is installed on your system.

---

**Figure 114  STUDENT AREA CODE INDEX ONLY panel**

| TRAN: DIVSTA | PFX SAMPLE APPLICATION SYSTEM | DATE: ::::::: |
| LTRM: ::::::: | (STUDENT AREA CODE INDEX ONLY) | TIME: ::::::: |
| MOD : PFXMOQA | | |

**STUDENT AREA CODE. . . ___**

**SELECT COURSE. . . . ___**

**LOCATION. . . . ___**

**STUDENT#. . . . ___**

(PRESS PF16 FOR STUDENT NAME)  **SSP: :::::**

<table>
<thead>
<tr>
<th>PHONE NO.</th>
<th>SURNMAE</th>
<th>FIRST NAME</th>
<th>COURSE</th>
<th>LOCN</th>
<th>STUDENT#</th>
</tr>
</thead>
<tbody>
<tr>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
</tr>
<tr>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
</tr>
<tr>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
</tr>
<tr>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
</tr>
<tr>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
</tr>
<tr>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
<td>::-::-::-::-:</td>
</tr>
</tbody>
</table>

ENTER=OPTION  PF12=CANCEL  PF13 THRU PF22 = OPTIONS 1 THRU A.

---

**STUDENT AREA CODE**

Type the student’s area code (three digits).
Online practice session

The online practice session uses the PFX-SAMP sample application. When you have completed all steps of the session, you can experiment with PFX-SAMP by adding, changing, and deleting segments. You can use PFX-SAMP whether or not you have Fast Path Indexer/EP installed on your system.

If you do not have Fast Path Indexer/EP installed on your system, you can still use the online practice session to become familiar with IMS Fast Path and actually perform some data entry and retrieval against DEDB databases.

If you do not have Fast Path Indexer/EP installed on your system, only some of the online practice session steps are available. Transactions DIVST6 through DIVST9 and DIVSTA are not available; therefore, you should not perform steps 8, 9, 10, 12, and 14.

**WARNING**

If you do not have Fast Path Indexer/EP installed on your system, using transactions DIVST6 through DIVST9 and DIVSTA will cause unpredictable results.

**To perform the online practice session**

1. Log on to IMS and ensure that all PFX-SAMP transactions, programs, and databases are available for use.

2. Begin the testing process.

   IMS: Type `/TEST MFS` and press Enter.

   Type `DIVST0` and press Enter. The PRIMARY MENU is displayed.
3 Select option 1 from the PRIMARY MENU and press **Enter**, or press **PF13** to access the COURSE MENU.

4 Select function G from the COURSE MENU and press **Enter**.

   The first course segment on the university database is displayed. To retrieve the next course, press **Enter**.

5 To add a course segment to the database, type the following values and press **Enter**:

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>COURSE NUMBER</td>
<td>12345</td>
</tr>
<tr>
<td>COURSE NAME</td>
<td>ABSTRACT ALGEBRA</td>
</tr>
</tbody>
</table>

   The acceptance message is displayed in the lower left corner of the panel.

6 To move to the LOCATION MENU, press **PF14**.

   The course number is displayed on this panel.

   To add a location segment for the newly entered course, type the following values and press **Enter**:

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>COURSE NUMBER</td>
<td>12345</td>
</tr>
<tr>
<td>LOCATION CODE</td>
<td>UCLA</td>
</tr>
<tr>
<td>LOCATION NAME</td>
<td>LOS ANGELES, CALIF.</td>
</tr>
</tbody>
</table>

7 To move to the INSTRUCTOR MENU, press **PF15**.

   The course number and location code are displayed on this panel.

   To add an instructor segment for the specified course and location, type the following values and press **Enter**:

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>COURSE NUMBER</td>
<td>12345</td>
</tr>
<tr>
<td>LOCATION CODE</td>
<td>UCLA</td>
</tr>
<tr>
<td>INSTRUCTOR ID</td>
<td>98765</td>
</tr>
<tr>
<td>INSTRUCTOR</td>
<td>SAM BROWNING</td>
</tr>
<tr>
<td>INSTRUCTOR SSN</td>
<td>591222333</td>
</tr>
<tr>
<td>PHONE NUMBER</td>
<td>416 921 2888</td>
</tr>
</tbody>
</table>
8 You must have Fast Path Indexer/EP installed to perform this step. To move to the ACCESS VIA INSTRUCTOR SSN panel, press PF20.

To access the instructor that you just added, type the following values and press Enter:

FUNCTION B
INSTRUCTOR SSN 591222333

9 You must have Fast Path Indexer/EP installed to perform this step. To change the instructor’s phone number and SSN, type the following values and press Enter:

FUNCTION C
SOC SEC # 0000000000
PHONE NUMBER 591 111 2222

10 You must have Fast Path Indexer/EP installed to perform this step. To browse this instructor, type the following values and press Enter:

FUNCTION B
SOC SEC # 0000000000

Because this index was built using sparse indexing to exclude instructors with SSNs equal to 000000000, the instructor segment is not in the secondary index.

11 To retrieve the instructor, press PF15 to return to the INSTRUCTOR MENU. Type the following values and press Enter:

FUNCTION B
COURSE NUMBER 12345
LOCATION CODE UCLA
INSTRUCTOR ID 98765

Enter a new SSN of 111111111 with a C function. It is rejected because the Instructor SSN index is a unique index, and that SSN belongs to another instructor on file.

12 You must have Fast Path Indexer/EP installed to perform this step. To access courses using the location code, press PF19. Then type UCLA and press Enter. All courses taught at this location are listed.

13 To add two students to the course and location, press PF16 to access the STUDENT MENU. Add the following students:
You must have Fast Path Indexer/EP installed to perform this step. To access the STUDENT NAME (INDEX ONLY) panel, press **PF21**.

Use this transaction to perform a name search. Type the student name **SMITH** and press **Enter**. All students with the name “Smith” will be listed. If you want more information about a specific student, enter that student’s course, location, and student numbers in the top portion of the panel and press. The STUDENT MENU for the selected student is displayed.

To display the ACCESS VIA STUDENT NAME panel, which is similar to the STUDENT NAME (INDEX ONLY) panel, press **PF18**. The ACCESS VIA STUDENT NAME panel retrieves information from the primary DEDB by using the secondary index database. ACCESS VIA STUDENT NAME also processes the Student Surname index in a stand-alone manner and uses the duplicate data of the secondary index database.

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>COURSE NUMBER</td>
<td>12345</td>
</tr>
<tr>
<td>LOCATION CODE</td>
<td>UCLA</td>
</tr>
<tr>
<td>STUDENT NUMBER</td>
<td>777777</td>
</tr>
<tr>
<td>SURNAME</td>
<td>SMITH</td>
</tr>
<tr>
<td>FIRST NAME</td>
<td>FRED</td>
</tr>
<tr>
<td>PHONE NUMBER</td>
<td>416 921 1489</td>
</tr>
<tr>
<td>SOC SEC #</td>
<td>345678912</td>
</tr>
<tr>
<td>ADDR1</td>
<td>ANYWHERE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>COURSE NUMBER</td>
<td>12345</td>
</tr>
<tr>
<td>LOCATION CODE</td>
<td>UCLA</td>
</tr>
<tr>
<td>STUDENT NUMBER</td>
<td>888888</td>
</tr>
<tr>
<td>SURNAME</td>
<td>SMITH</td>
</tr>
<tr>
<td>FIRST NAME</td>
<td>GEORGE</td>
</tr>
<tr>
<td>PHONE NUMBER</td>
<td>416 222 3333</td>
</tr>
<tr>
<td>SOC SEC #</td>
<td>545678912</td>
</tr>
<tr>
<td>ADDR1</td>
<td>SOMEWHERE</td>
</tr>
</tbody>
</table>
Index

A

ACB library 159
ACBGEN does not support PROCSEQ 102
ACBGEN Utility report 87
ACCESS keyword 60, 61
ACCESS VIA INSTRUCTOR SSN (DIVST8) 240
ACCESS VIA INSTRUCTOR SSN Panel 240
ACCESS VIA LOCATION CODE (DIVST7) 238
ACCESS VIA STUDENT NAME (DIVST6) 237
adding a PFX index 99
AJ status code 46, 49
AK status code 47
alternate processing sequence 41, 42, 101
AMODE keyword 111
APF authorization 127
application processing restrictions
  non-retrieval processing using PROCSEQ 51
SENSEG order 42
areaname DD statement 145
attributes
  reentrant (RENT) 111, 117
  reusable (REUS) 111, 117
AU status code 39, 46

B

BMC Software, contacting 2
BMC Software-supplied front-end process 82
BMCRRC00 module 82
BMP mode
  index resychronize function 175
BMP mode JCL 134
BMP mode processing 126, 133
Boolean operators 46
BUILD command 154
BUILD control statement 86, 198
build DBD source 198
build PF definition block 70
Build Summary Report 155

C

caching
  index build function 162
  index verify function 185

D

D command code 45, 50
DBD
  build source 198
  maintenance 99
update DEDB source DBDs 106
update DEDB source DBDs only 107
updates that affect only DEDB source 107
updates that affect PFX indexes 106
DBD keyword 86, 144, 157, 180
DBD processing under PFXAGEN1 79
DBRC 33
  during index function 155
  during index verify 177
DBRC keyword 130
DBRC region services 136, 140
DBRC use 130
DD statements
  DFSRESLB 83
EXEC 82
IMS 83
IMSACB 83, 129, 140
MODSTAT 129, 140
OLCSTAT 129, 140
PFXLIB 83
PFXPRINT 84
STEPLIB 83
SYSIN 83
SYSLIB 83
SYSLIN 83
SYSPRINT 84
SYSUT1 84
WORK001 84
DDATA 68
defining index structure
SHISAM or HISAM 60
DELETE control statement 86
delete index registration 105
DFSMDA macro sample 92
DFSMDA member 130
DFSRESLB DD statement 83
DFSRRC00 member 127
DFSRRC00 module 82
JCL example 82
diagnostic tools 203
disassemble
DBDs 194
PSBs 194
DMB 129, 140
DSILIB Online Change utility
flow diagram 93
JCL example 94
duplicate data 53
dynamic allocation 128, 130, 139
DD statements 90
DFSMDA macro sample 92
JCL example for PFXAGEN 89
registration DD statements 90
temporary DD statements 91
EXTRTN keyword 68

F
Fast Path Indexer/EP macros
PFXCHILD 60, 161
PFXGEN 60, 70
PFXPART 60, 63
PFXXDFLD 60, 67, 162
Fast Path Indexer/EP reports
ACBGEN Utility report 87
Build Summary Report 155
Resynchronization Summary Report 177
Verify Summary Report 176
Fast Path Indexer/EP status codes 44
FIELD macro statement 67
formats, syntax diagrams 22

G
GB status code 62, 115
GE status code 42

H
Help, online 19
hierarchy, inversion of 42
HISAM index
symbolic pointer 39
HISAM index structure 34

I
IAREA keyword 138, 144, 157, 180
ICACHE keyword 148, 162, 185
IFP mode
index XSCAN function 143
IFP mode JCL 137
IFP mode processing 136
IMS ACB library 129, 140
IMS DD statement 83
IMS RESLIB 34
IMS transaction codes
DIVST0 - PRIMARY MENU 228
DIVST1 - COURSE MENU 229
DIVST2 - Location Menu 230
DIVST3 INSTRUCTOR MENU 232
DIVST4 - STUDENT MENU 234
DIVST5 - PREREQUISITE MENU 236
DIVST6 - ACCESS VIA STUDENT NAME panel 237, 238, 240, 241, 243
DIVST7 - ACCESS VIA LOCATION CODE panel 238
DIVST8 - ACCESS VIA INSTRUCTOR SSN panel 240
DIVST9 Student Name Index Only panel 241

ECSA 111, 117
electronic documentation 19
error queue elements (EQEs) 156, 178
examples
DFSMDA macro 92
partitioned index randomizing exit routine 117
PCB for primary database with PROCSEQ 102
PCB primary database without PROCSEQ 102
PCB, secondary index 103
primary database 40
Secondary Index Print/Verify utility 197, 198
EXEC DD statement 82
exit routine 114
exit routines
structure 112, 113
EXTRTN keyword 110
extended common service area 111, 117
DIVSTA Student Area Code Index Only panel 243
IMS Transaction Codes to PF Key Table 225
IMS/VS commands 136
IMSACB DD statement 83, 129, 140, 159
index
  add a partitioned index randomizing exit routine 114
  add a sparse indexing exit routine 109
  add secondary index database maintenance exit routine 109
  adding a PFX index 99
  delete index registration 105
  duplicate data 53
  maintenance 98
  stand-alone database 53
  structure 34
  subsequence fields 39
  update DEDB source DBDs 106
  update PSBs 108
index BUILD command 124
index build function 124
  caching 162
  keywords and subcommands 156
  preparation 154
  scenarios 166
index database examples
  PCB for inverted full-function database 43
  PCB for inverted primary DEDB 43
  primary database with inverted hierarchy 43
  SSA that could be used with inverted hierarchy 44
index database maintenance exit routine
  applying maintenance 113
  coding considerations 111
  maintenance functions 110
  parameter list for user-written exit 112
  registers at entry to user-written exit 111
  registers at time of return from user-written exit 112
  single-character method 110
  structure for user-written exit 112, 113
  user-written method 110
INDEX keyword 146, 182
index partitioning 160
  lateral sequential processing 161
  vertical sequential processing 161
index recovery 33
index registration process 55
  add PFX macros to the source DBD 99
  create or update PSB source 101
  flow diagram 75
  generate index DBD source 101
  run IMS PSBGEN 103
index RESYNC command 125
index resynchronize function 125
  preparation 175
  scenarios 190
index segments
  root segment 39
  SHISAM or HISAM 39
  source segment 38
  target segment 38
index structure 30
  HISAM 34
  SHISAM 34
index VERIFY command 125
index verify function 125
  caching 185
  keywords and subcommands 178, 179
  scenarios 188
index XSCAN command 124
index XSCAN function 124
  caching 148
  keywords and subcommands 143
  scenarios 149
INDEX_THREADS keyword 163, 186
INPUT_DSN_MASK keyword 130
INPUT_THREADS keyword 162, 185
installation verification procedures 217
INSTRUCTOR MENU (DIVST3) 232
inverted hierarchy 42, 43
IX subcommand 145, 182

J

JCL
  BMP mode 134
  IFP mode 137
  offline mode 127
JCL examples
  DFSRRC00 82
  DSILIB Online Change utility 94
  PFXAGEN using dynamic allocation 89
  PFXAGEN utility 82
  PFXLIST utility 195
JCL requirements
  PFXAGEN utility 82
  PFXLIB Online Change utility 94
  PFXLIST utility 195
journal database 224

K

key searches
  complete 48
  partial 49
  using a PCB with a secondary index 52
  using a PCB with PROCSEQ 51
  using a PCB without PROCSEQ 50
keywords
  AMODE 111
  DBD 144, 157, 180
  DBRC 130
  EXITRTN 110
  IAREA 138, 144, 157, 180
PSB processing under PFXAGEN utility 80
relationship to ACBLIB 78
PFXLIB DD statement 83
PFXLIB Online Change utility
JCL requirements 94
PFXLIST utility
control statement syntax 197
control statements 197
control statements and keywords 198
DD statements 196
inputs and outputs 194
JCL example 195
JCL requirements 195
report 199
PFXLIST utility control statements
BUILD 198
LIST 198
PFXLIST utility report 199
PFXPART macro 60, 63
examples 65
NAME keyword 64
syntax diagram 64
PFXPRINT DD statement 84
PFXDFLD macro 60, 67, 162
syntax diagrams 69
PFXDFLD macro keywords
DDATA 68
EXTRTN 68
NULLVAL 68
PARTEXIT 69
SEGMENT 69
SRCH 67
SUBSEQ 68
Phonetic Surname index 223
PREREQUISITE MENU (DIVST5) 236
Primary 226
primary database example 40
primary DEDB 220
primary DEDB transaction instructions
add 226
browse 227
change 226
delete 226
get 227
get next within parent 227
PRIMARY MENU (DIVST0) 228
primary processing sequence 102
processing modes 126
PROCSEQ keyword 35, 41, 76, 89, 102, 108
product changes 24
product support 3, 205
PSB
retrievals 41
update 108
update DBDs that affect indexes 107
PSB keyword 86
PSB processing under PFXAGEN utility 80
PSB source retrieval
alternate processing sequence 101
primary processing sequence 102
PSBSRC data set 89
PSBSRC DD statement
include during registration process 103
PST keyword 111
publications, related 20
R
RECON data sets 131, 136, 140
recovery 33
recovery status 131
reentrant (RENT) attribute 111, 117
REGION size 127, 134, 138
registered area data set 130
registers at entry to user-written exit 111
registers at time of return from user-written exit 112
Registration Verification Report 199
related publications 20
reports
ACBGEN Utility report 87
Build Summary Report 155
Registration Verification Report 199
Reysnchronization Summary Report 177
Verify Summary Report 176
RESYNC command 175
Resynchronization Summary Report 177
return codes for PFXAGEN utility 86
reusable (REUS) attribute 111, 117
RMODE keyword 111
S
secondary index databases 222
Secondary Index Print/Verify utility
examples 198
flow diagram 195
secondary index, PCB example 103
SEGMENT keyword 69
SENSEG 102
SENSEG order 42
sensitive segment 42, 102
shadow PCB 76
added under PFXAGEN utility 80
SHISAM index
symbolic pointer 39
SHISAM index structure 34
SHISAM index with subsequence fields 224
SORT keyword 149, 164, 187
SORT_OPTION keyword 187
SRCH keyword 67
SSASIZE keyword 46
stand-alone database 53
status codes 44
   AJ 46, 49
   AK 47
   AU 39, 46
   GB 62, 115
   GE 42
   NE 45
   NI 45
STEPLIB DD statement 83
structure of index 34
STUDENT AREA CODE INDEX ONLY Panel (DIVSTA) 243
STUDENT MENU (DIVST4) 234
STUDENT NAME INDEX ONLY Panel (DIVST9) 241
subcommands
   IX 145, 182
SUBSEQ keyword 68
subsequence fields 39, 46
summary of changes 24
support, customer 3
symbolic pointer
   HISAM index 39
   SHISAM index 39
syntax rules 197
syntax statement conventions 21
syntax, format for diagrams 22
SYSSIN DD statement 83
SYSLIB DD statement 83
SYSLIN DD statement 83
SYSPRINT DD statement 84
SYSUT1 DD statement 84

T

technical support 3

U

update DEDB source DBDs 106
using DBRC 130

V

VERIFY command 174
Verify Summary Report 176
vertical sequential processing 115

W

WORK001 DD statement 84
Notes