KMSYS
WORLDWIDE, INC

Database Reorganization Utility Reference

10 August 2011

The information contained in this document is the latest available at the time of preparation; therefore, it may be changed without notice, and it does not represent a commitment on the part of KMSYS Worldwide, Inc. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy this software on magnetic tape, disk, or any other medium for any purpose other than stated in the terms of the agreement, or without the express written permission of KMSYS Worldwide, Inc.

If you have any comments about the software or documentation, notify KMSYS Worldwide, Inc., in writing at the following address:

KMSYS Worldwide, Inc.
P.O. Box 669695
Marietta, Georgia 30066
U.S.A.

Technical Support (770) 635-6363 - Main Number (770) 635-6350 - Fax (770) 635-6351

I-QU PLUS-1 Release 11R6, November 1999

Table of Contents

Chapter 1: Introduction

This manual will describe the database reorganization related commands of the I-QU PLUS-1 processor, and the standalone I-QU PLUS-1 Database Utility programs used to perform various tasks relating to database reorganization. The utility programs are used in conjunction with the I-QU PLUS-1 Processor.

For a more complete description on how these programs are used in database reorganization, see the I-QU PLUS-1 Reorganization User Guide.

1.1 Description of Chapters

Chapter 1 contains a brief introduction, a summary of each chapter and the syntax notation used within this manual.

Chapter 2 describes the special I-QU PLUS-1 commands used to manipulate pointers when performing database reorganizations.

Chapter 3 describes the Query Schema Utility (QRYSCH) used to present schema information in 80-column format suitable for viewing online. In addition, an SGS output option is fully illustrated.

Chapter 4 lists the directives used to alter a working copy of a schema file with the Schema Modification Utility (SCHUTL). A working copy is frequently modified to improve load performance or when reorganizing TIP areas assigned as EXEC files.

Chapter 5 lists the directives used to replace record set pointers, unlink sets and globally change database pointer area/page/slot partitions using the Pointer Correction Utilities (PBLD & PFIX).

1.2 Syntax Notation

The following conventions are used throughout this manual in the description of I-QU PLUS-1 commands:

- Changes to this document since its last publication are marked with a change bar (an elongated vertical bar) as shown to the right of this paragraph.
- Important notes and warnings are encased in a box as shown around this bullet.
- All words in UPPERCASE letters (not italicized) are reserved keywords and must be entered exactly as shown or in their abbreviated forms.
- Many keywords may be abbreviated. Abbreviations will be highlighted in BLUE throughout the syntax shown in this manual. For example, DEFINE may be abbreviated as "DEF".
- All *italicized* words (mostly in lowercase letters) are to be substituted by a user supplied name or value.

- Ellipsis (…) implies allowable, but omitted, repetitions in the published syntax. Please note that the ellipsis is ***not*** allowed in the command or directive when parsed. In the following example, multiple set names may be specified:

  set-name-1 [… *set-name-n*]

- A vertical bar (|) represents an "or" or "and/or" operator.
- Selections appearing within brackets, "[ ]", are lists of optional items of which one may be selected. In the following example, neither A nor B is required, but either one or the other may be selected:

  [A | B]

- An underlined word in optional brackets represents the default value when not entered. For example, in the syntax, "[RECORDS | CHARACTERS]," RECORDS is the default and would be assumed if omitted.
- Selections appearing within braces, "{ }", are lists of items of which one and only one must be selected. In the following example, one of either C, D or E must be selected:

  {C | D | E}

- Selections appearing within double vertical bars, "||", are lists of items of which one or more must be selected. The items between vertical bars are referred to as permutations and may be selected in any order. In the following example, one or more of F, G, and/or H must be selected:

  ||F | G | H||

Chapter 2: Pointer Manipulation Commands

The following I-QU PLUS-1 Processor directive and commands should only be used by persons familiar with DMS 2200 page and record structures. Pointer commands and directives are available only when I-QU PLUS-1 has been generated for both SINGLE-THREAD and POINTER MANIPULATION.

Pointer manipulation commands have been designed to be used with the database reorganization package, and have been tailored accordingly. Refer to the I-QU PLUS-1 Database Reorganization User Guide for additional information concerning pointer manipulation commands and more examples of their usage.

2.1 Pointer Area Definition Directive

Pointer area definition is available only when the I-QU PLUS-1 Processor has been generated for single-thread with the optional pointer manipulation commands. The purpose of this definition is to specify a work area for database pointers retrieved from records within the DMR's page buffer. The user must specify the number of words to be reserved for pointers. A pointer definition may also be referenced as an ASCII alphanumeric string variable, in which case its character length would be used (4 times its word length).

Format:

DEFINE P *pointer-work-area number-of-words*

DEFINE may be abbreviated DEF.

Example:

```
DEFINE P PTR-HOLD 12
```

This directive will reserve an area of 12 words (or 48 characters).

2.2 GETPTR

The GETPTR command is used to retrieve a contiguous set of database pointers from the record that is current of run unit. I-QU PLUS-1 will locate the page and record within the DMR page buffer to accomplish this. If the user specifies pointer positions that are not actually a part of the control portion of the record, a run-time error will result. Pointers will be moved beginning with the first pointer specified from the record to the first word of the pointer work area.

Format:

GETPTR (*start*-word-in-record,*number-of-pointers*) *pointer-work-area*

GETPTR may be abbreviated GETP.

If *start-word-in-record* is zero, the value in the reserved variable, S$, will be substituted at execution time. If *number-of-pointers* is zero, the value in the reserved variable, L$, will be substituted at execution time.

Example:

```
DEFINE P P-HOLD 5                    .  Define pointer work area
FETCH3 N PRODUCT-REC PRODUCTS AREA
GETPTR (1,3) P-HOLD                  .  Get first three pointers
```

2.3 INSPTR

The INSPTR command is used to insert new set pointers into the pointer work area. Existing pointers will be shifted right to accommodate the new pointers. Words shifted beyond the right limit of the pointer work area will be dropped. The new pointers will be given the value of the specified database pointer variable.

Format:

INSPTR (*start-word-in-pointer-area*,*number-of-pointers*) ;
pointer-work-area DBP-variable

INSPTR may be abbreviated INSP.

If the *start-word-in-pointer-area* is zero, the value in the reserved variable, S$, will be substituted at execution time. If *number-of-pointers* is zero, the value in the reserved variable, L$, will be substituted at execution time.

DBP-variable must be numeric.

Example:

```
INSPTR (2,1) P-HOLD SAVE-DBP
```

All pointers following word 2 in the pointer work area will be shifted right one position and the value of variable SAVE-DBP will be put into vacated word 2.

2.4 NULPTR

The NULPTR command is used to mark all empty set pointers in a pointer work area as NULL. When records are replaced during the reload phase of reorganization, empty set pointers can thus be recognized. Empty set pointers will then be set to the value of the record's new database pointer using the RSTPTR command. This command will inspect all words in the pointer work area and mark any word that is equal to the value in the specified database pointer variable. (The NULL value is 077777777777 octal, not -0).

Format:

NULPTR *pointer-work-area DBP-variable*

NULTPR may be abbreviated NULP.

Example:

```
NULPTR P-HOLD C-DBP
```

Mark as NULL any word in P-HOLD that matches the value of the reserved variable C-DBP.

The NULPTR/RSTPTR commands should not be used if the owner of a set is to be PFIXed.

2.5 PUTPTR

The PUTPTR command is the reverse of the GETPTR command. It is used to put pointers into the record that is current of run unit. Pointers will be moved from left to right from the first word of the pointer work area to the specified position in the record.

Format:

PUTPTR (*start-word-in-record*,*number-of-pointers*) *pointer-work-area*

PUTPTR may be abbreviated PUTP.

If the *start-word-in-record* is zero, the value in the reserved variable, S$, will be substituted at execution time. If the *number-of-pointers* is zero, the value in the reserved variable, L$, will be substituted at execution time.

Example:

```
PUTPTR (2,2) P-HOLD
```

This command will move the first two positions from P-HOLD into pointer positions 2 and 3 of the current record.

2.6 REMPTR

The REMPTR command is used to remove set pointers from the pointer work area. Pointers to the right will be shifted left to replace removed pointers. Words vacated at the right of the pointer work area will be changed to binary zeroes.

Format:

REMPTR (*start-word-in-pointer-work-area*,*number-of-pointers*) *pointer-work-area*

REMPTR may be abbreviated REMP.

If the *start-word-in-pointer-work-area* is zero, the value in the reserved variable, S$, will be substituted at execution time. If the *number-of-pointers* is zero, the value in the reserved variable, L$, will be substituted at execution time.

Example:

```
REMPTR (1,2) P-HOLD
```

Pointers following word 3 will be shifted two positions to the left overlaying pointers in words 1 and 2.

2.7 RSTPTR

The RSTPTR command is the reverse of the NULPTR command. It is used to convert NULL set pointers back to database pointers. The command will inspect each word of the pointer work area and replace any word containing a NULL pointer value with the value contained in the database pointer variable.

Format:

RSTPTR *pointer-work-area DBP-variable*

RSTPTR may be abbreviated RSTP.

Example:

```
RSTPTR P-HOLD C-DBP
```

This command will set NULL pointers to the value of the reserved variable C-DBP.

The NULPTR/RSTPTR commands should not be used if the owner of a set is to be PFIXed.

2.8 SET CURRENT DBP

This command is similar to the SET CURRENT DBK command, except that the current record's database pointer will be moved to the reserved variable C-DBP.

Format:

[SET] CURRENT DBP

CURRENT may be abbreviated CURR.

2.9 SETPTR

The SETPTR command is used to set specified words of the pointer work area to the value in the specified database pointer variable. No word shifts will occur.

Format:

```
SETPTR  (start-word-in-pointer-work-area,number-of-pointers) ;
   pointer-area  DBP-variable
```

SETPTR may be abbreviated SETP.

If the *start-word-in-pointer-work-area* is zero, the value in the reserved variable, S$, will be substituted at execution time. If the *number-of-pointers* is zero, the value in the reserved variable, L$, will be substituted at execution time.

Example:

```
SETPTR (2,2) P-HOLD MY-DBP
```

This command will set words 2 and 3 of the pointer work area to the value in the user variable MY-DBP.

2.10 XREF

The XREF command is used to cross-reference a new database record location to its old database location (used when reloading records during reorganization). The new database pointer must be in the second DBP-variable. I-QU PLUS-1 will write the old-DBP/new-DBP cross-reference record to file DBPFILE. This file must be @ASGed and @USEd before this command is executed. I-QU PLUS-1 will automatically open and close the cross-reference, which is built as a result of the XREF command. If the ALT option is used, the output file ALTDBPFILE will be assumed @ASGed and @USEd.

Format:

XREF *old-DBP/DBK-variable new-DBP/DBK-variable* [ALT]

XREF may be abbreviated XR.

xample:

Create a cross-reference of OLD to current database pointer for the current record.

```
XREF HOLD-OLD-DBP C-DBP
```

Create a cross-reference of OLD to current database key for the current record.

```
XREF HOLD-DATABASE-KEY C-DBK ALT
```

2.11 XTRPTR

The XTRPTR command is used to extract a single pointer word from the contents of a pointer work area and place it in a defined numeric variable, thus allowing the user to isolate and manipulate it.

Format:

XTRPTR (*word-in-pointer-work-area*) *pointer-work-area DBP-variable*

XTRPTR may be abbreviated XTRP.

If the *word-in-pointer-work-area* is zero, the value in the reserved variable, S$, will be substituted at execution time.

Example:

```
GETPTR (1,9) H-PTR-----. Get pointers from record
XTRPTR (3) H-PTR X-----. Put pointer 3 into X
BITSPLIT X INTO X,Y,Z (10,17,9)  . Split into area/page/slot
```

Chapter 3: QRYSCH - Query Schema Utility

QRYSCH is an interactive program that allows the user to access and display information contained with tables in the object schema. The tables that may be displayed are:

RDT	Record Description Table
RRT	Record Reference Table
SDT	Set Description Table
ART	Area Reference Table
ALR	Displays all above, except RRTs for entire schema.

The output of an RDT query includes information such as: which area or areas an object record resides in; which sets the record participates in as an owner or member; the position of the manual control word and manual set pointers; and the break-down of area, page, and slot bits in database pointers. An SDT query includes information pertaining to which records participate in a set. An ART query shows information on the size, type and mode of an area. The RRT contains limited information. It is generally not referenced for planning reorganizations.

QRYSCH is executed as follows:

```
@QRYSCH,options
```

The QRYSCH processor name used will depend on the installation mode used when I-QU PLUS-1 was installed. The default is QRYSCH. See the section in the I-QU PLUS-1 Installation Guide called "The Product Files" for more information on the naming conventions used for processors.

There are five options available. They are:

- B Causes the execution of QRYSCH to be treated as a batch mode execution.
- D Causes the execution of QRYSCH to be treated as a demand mode execution. This option is the converse of the B-Option.
- H The H option causes the help screen to be displayed initially.
- S Causes an SGS file to be output as file name "SGSFILE". This file must be currently assigned to the run. The S option causes all RDTs, SDTs and ARTs in the schema to be displayed and output in SGS format.
- T For KMSYS Worldwide debugging only. Use only if directed to by KMSYS Worldwide' personnel.

QRYSCH will prompt the user for the object schema file name, table type and area, record or set name. If a question mark is entered at the prompt, a help screen describing the input parameter syntax will be displayed.

The following is a sample QRYSCH session:

```
@QRYSCH <—— User executes program
QRYSCH 11R6 (Release 11R6) (991107 1500:37) 1999 Nov 18 Thu 1247:25
(C) Copyright 1983-1996 by KMSYS Worldwide, Inc.  All Rights reserved.
This program licensed for use by KMSYS Worldwide, INC.
Enter request(Q*F.SCHEMA,TBL,REC) or "END" to term., or "?" for help:
DMS*SCHABS.DEMOSCHEMA,RDT,ORDER-HEADER-REC

================================================================================
RDT information for: ORDER-HEADER-REC                 record code:000011
Location mode is CALC, DUPS. NOT ALLOWED
Total record len. (data)..  43 Control portion length....    8
Owner of sets.............   2 Member of auto. sets......    1
Member of man. sets.......   1 Reserved man. ptrs........    1
Offset to man. ctl. word..   5
________________________________________________________________________________
Within ORDERS                           code:   4
    *** SARP ***
    Area bits:10, page bits:17, rec/slot bits: 9
Set ownership information:

  ORDH-LINE                         code:   7 #ptrs.= 2 ptr.pos.=  1
  ORDH-CMT                          code:   8 #ptrs.= 1 ptr.pos.=  3
________________________________________________________________________________
Automatic set membership information:

  CUST-ORD                          code:  27 #ptrs.= 1 ptr.pos.=  4
________________________________________________________________________________
Manual set membership information:
(bits numbered 1 thru 36, left to right)

  ORDER-OPEN                        code:  30 #ptrs.= 1 MCW bit pos.= 1
   Manual control word mask for relink = 1
Key information:
Key: 1 defined as: (  12,   7) DISP
DEFINE C key-def-n (CCCSWWWW) for Key:  1  (00740002)
________________________________________________________________________________
<<<No WITHIN clause for this record.>>>
* End of RDT display *
Enter request(Q*F.SCHEMA,TBL,REC) or "END" to term., or "?" for help:

,,,SDT,ORDH-LINE
================================================================================
SDT Information for:ORDH-LINE                         set code:000007
Set mode is CHAIN LINKED PRIOR
Order is: SORTED BY DEFINED KEYS
Owner record is: ORDER-HEADER-REC                code: 000011
Members are:
  ORDER-LINE-REC                  type:COS     code:  14 #ptrs.= 2 ptr.pos.= 2
Key: 1 defined as: (   1,   6)DISP ASCENDING
<<< End of SDT display >>>
________________________________________________________________________________
Enter request(Q*F.SCHEMA,TBL,REC) or "END" to term., or "?" for help:

,,,ART,ORDERS

Area: ORDERS       code: 000004
   Area mode is DATA.
   Allocated pages:000500
   *** SARP ***
   Area bits:10, page bits:17, rec/slot bits: 9
 Allocate     50 overflow pages AT END
```

```
Page length:  896 words
Number of CALC chains:   1
Load Factor: 100 percent
_____________________________________________________________________________
Enter request(Q*F.SCHEMA,TBL,REC) or "END" to term., or "?" for help:
CSF:@FREE I$QUSCH.  .

**** End of schema query ***
```

In the above example, QRYSCH was run in demand mode. QRYSCH may be executed in batch mode to obtain printed results. The following is an example of a QRYSCH batch run to create the SGS file and display information on the entire schema.

```
@RUN,A DBA01,,IQU
@ASG,UP SGSFILE.
@QRYSCH,S
DMS*SCHEMAFILE.MAINSCHEMA
@FIN
```

The SGS file output from QRYSCH may be used by the local user site in the creation of SSG skeletons to generate database reorganization runs.

The following is the format of the various SGS images that may be produced by QRYSCH (labels and key words are indicated by uppercase letters):

SCHEMA *qualifier*,*filename schema-name*

RECORD *record-name record-code* ;
record-data-word-length control-portion-word-length ;
owner-of-sets auto-member-of-sets man-member-of-sets ;
word-offset-to-manual-control-word reserved-pointers ;
location-mode ;
[INDEXED-AREA,*area-name*[, …] ;]
AREAS,*area-name*[, …] ;
OWNERSETS,*set-name*,*set-code*,*number-pointers*,*pointer-position*[, …] ;
MEMAUTOSETS,*set-name*,*set-code*,*number-pointers*,*pointer-position*[, …] ;
MEMMANSETS,*set-name*,*set-code*, ;
number-pointers,*MCW-bit-position*,*MCW-mask-for-relink*[, …] ;
KEY,*starting-byte-position*,*byte-length*,*data-type* ;
[,KEYDEF,*key-number*,(*cccswwww*)[, …] ;]
AREALIMITS,*area-code*,*lower-bounds*,*upper-bounds*

SET *set-name set-code owner-rec-name owner-rec-code* ;
[SORT-SEQ,*starting-byte-position*,*byte-length*,*data-type*,*sort-seq*[, …] ;]
MEMBERS,*member-rec-name*,*member-rec-code*, ;
number-of-pointers,*pointer-position/MCW-bit-position*,*SOS-type*[, …]

AREA *area-name area-code allocated-pages* ;
area-mode area-type area-bits page-bits slot-bits ;
page-size load-factor calc-chains ;
interspersed-data interspersed-overflow at-end-overflow

The KEYDEF subfield on the KEY field above is only generated for CALC records. The parameter, (*cccswwww*), can be used on I-QU PLUS-1's DEFINE C directive. See the I-QU PLUS-1 Programmer Reference.

When the *lower-bounds* or the *upper-bounds* on the AREALIMITS SGS is a database data name, "D-NAME" will appear in the SGS instead of the page number.

The following is an example of an "SGSFILE" built using QRYSCH:

```
SCHEMA DMS,SCHABS CLASS
```

```
RECORD CUST-KEY-REC 2 12420000 CALC ;
    AREAS,CUSTOMER ;
    OWNERSETS,CUST-ADDR,5,1,1,CUST-ORD,27,1,2 ;
    KEY,1,9,DISP,KEYDEF,01,(00910000)

RECORD CUST-ADDR-REC 4 45301000 ISAM ;
    INDEXED-AREA,CUST-ADDRX ;
    AREAS,CUST-ADDRESS ;
    MEMAUTOSETS,CUST-ADDR,5,1,1 ;
    KEY,1,29,DISP

RECORD ORDER-HEADER-REC 11 43821151;
         CALC ;
    AREAS,ORDERS ;
    OWNERSETS,ORDH-LINE,7,2,1,ORDH-CMT,8,1,3 ;
    MEMAUTOSETS,CUST-ORD,27,1,4 ;
    MEMMANSETS,ORDER-OPEN,30,1,1 1 ;
    KEY,12,7,DISP,KEYDEF,01,(00740002)
RECORD ORDER-CONTROL-REC 125610022;
         CALC ;
    AREAS,ORDCTL ;
    OWNERSETS,ORDER-OPEN,30,1,1 ;
    KEY,1,5,DISP,KEYDEF,01,(00510000)
RECORD ORDER-COMMENT-REC 13 21201000;
         VIA,ORDH-CMT ;
    AREAS,ORDERS ;
    MEMAUTOSETS,ORDH-CMT,8,1,1
RECORD ORDER-LINE-REC 14 18301000;
         VIA,ORDH-LINE ;
    AREAS,ORDERS ;
    MEMAUTOSETS,ORDH-LINE,7,2,1
SET CUST-ADDR 5 CUST-KEY-REC 2 ;
    MEMBERS,CUST-ADDR-REC,4,1,2,COS
SET ORDH-LINE 7 ORDER-HEADER-REC 11 ;
    SORT-SEQ,1,6,DISP,ASCENDING ;
    MEMBERS,ORDER-LINE-REC,14,2,2,COS
SET ORDH-CMT 8 ORDER-HEADER-REC 11 ;
    MEMBERS,ORDER-COMMENT-REC,13,1,2,COS
SET CUST-ORD 27 CUST-KEY-REC 2 ;
    MEMBERS,ORDER-HEADER-REC,11,1,5,COS
SET ORDER-OPEN 30 ORDER-CONTROL-REC 12 ;
    MEMBERS,ORDER-HEADER-REC,11,1,1,COS
AREA CUST-ADDRESS 1 50 DATA SARP 10 17 9 896 10010010
AREA CUST-ADDRX 2 5 INDX SARP 10 17 9 896 1001000
AREA CUSTOMER 3 50 DATA SARP 10 17 9 896 1001005
AREA ORDERS 4 500 DATA SARP 10 17 9 896 10010050
AREA ORDCTL 5 100 DATA SARP 10 17 9 896 10010010
```

Chapter 4: SCHUTL - Schema Modification Utility

SCHUTL is a program used to perform certain modifications to an object schema (and subschema) without incurring the overhead of the DDL and SDDL processors. SCHUTL will directly modify the object schema and subschema absolute element upon execution. Using the SCHUTL directives allows the user to produce special schemas and subschemas for load efficiency. The following changes can be made with SCHUTL:

- Change the TIP file number or EXEC file name in order to use an alternate schema file.
- Eliminate LOOKS to eliminate I/Os to the recovery files.
- Set all areas to NON-PREINIT, thus eliminating the need to initialize the area with the DMU.
- Reduce the LOAD factor to allow room on pages to store records during update.
- Change set order to NEXT to avoid the overhead (I/Os) associated with set ordering during reload.
- Ignore the linking of records into secondary sets during reload, thus eliminating the I/Os otherwise required to the owner/members of the secondary sets. Note: the PFIX utility provides the mechanism to re-link these sets.
- Ignore RESULT processing on owners when reloading members, thus eliminating I/Os to the owners.
- Change set occurrence selection from LMO to COS, thus eliminating expensive navigational I/Os otherwise required to store members.

Changes made by SCHUTL are permanent; therefore, schemas and subschemas to be modified should be backed up to an alternate EXEC file before using this program.

To use SCHUTL, call the processor as follows:

```
@SCHUTL, options
Followed by SCHUTL Directives
@EOF
```

The SCHUTL processor name used will depend on the installation mode used when I-QU PLUS-1 was installed. The default is SCHUTL. See the section in the I-QU PLUS-1 Installation Guide called "The Product Files" for more information on the naming conventions used for processors. The options are:

B Causes the execution of SCHUTL to be treated as a batch mode execution.

D Causes the execution of SCHUTL to be treated as a demand mode execution. This option is the converse of the B-Option.

T For KMSYS Worldwide debugging only. Use only if directed to by KMSYS Worldwide' personnel.

4.1 USE SCHEMA Directive

The USE SCHEMA must be the first SCHUTL directive. It specifies the name of the object schema to be modified and the name of the file in which it will be found.

Format:

USE SCHEMA *schema-name* FILE *file-name*

4.2 CHANGE ALL TIP Directive

This directive allows the area names to be used as EXEC file names while the TIP file numbers are ignored. The change is made to all areas in the schema and the schema file. If the schema/subschema file is TIP, the CHANGE EXEC FILE NAME directive must be used as no file name is placed in the absolute when the schema/subschema is compiled.

Format:

CHANGE ALL TIP TO EXEC

4.3 CHANGE AREA LOOKS Directive

This directive will set the type of looks (ART40) in the area reference table for each area in the object schema to NO LOOKS. This directive is generally used so that I-QU PLUS-1 may be run single-thread without the overhead of QUICK-LOOK file I/O.

Appropriate recovery measures must be taken prior to any destructive DML or pointer manipulation operations being performed.

Format:

CHANGE AREA LOOKS TO NOLOOKS

4.4 CHANGE AREAS Directive

This directive changes all areas in the object schema to be NON-preinitialized, thereby eliminating the need to run a DMU initialize before starting an area reload.

Format:

CHANGE AREAS TO NON-PREINIT

4.5 CHANGE DUPS TO ALLOWED ON CALC Directive

The CHANGE DUPS TO ALLOWED ON CALC directive will change the schema to allow CALC records to be loaded as if the "DUPLICATES ALLOWED" clause were used in the schema for the named record. This eliminates the need for the DMR to search the entire CALC chain when storing a record. For CALC areas where off-page chaining is anticipated, this change can result in considerable I/O savings.

Format:

CHANGE DUPS TO ALLOWED ON CALC RECORD *record-name*

4.6 CHANGE EXEC FILE NAME Directive

The CHANGE EXEC FILE NAME directive will change the file name of the current schema (named in the USE SCHEMA) as well as the specified subschema. The host language of the subschema used here may only be ASCII COBOL.

The CHANGE EXEC FILE NAME directive is used to alter the file name in the object schema and subschema. This alteration enables I-QU PLUS-1 to use a schema and subschema from

an alternate EXEC file without affecting the schema in the production EXEC file. Using this directive in combination with other SCHUTL directives allows the user to produce special schemas and subschemas for load efficiency. The absolute schema and subschema must be copied to the specified alternate EXEC file prior to being INVOKED by the I-QU PLUS-1 Processor.

Format:

CHANGE EXEC FILE NAME *subschema-name* TO [*qualifier**]*filename*

4.7 CHANGE LOAD FACTOR Directive

The LOAD factor allows you to leave space on a page for records that may be added after the load. Reserving space in this manner may be useful for volatile areas. For example, using a LOAD factor can facilitate storing new members on the same page as the owner or other members in a set occurrence. It can also be used to disperse Index Sequential or CALC entry point records throughout an area during load, thus leaving room for the placement of additional records on prime pages during update. If space is not reserved and a page is full, the DMR placement strategy must select another data or overflow page. Consequently, the search may be through pages, which are already full.

Format:

CHANGE LOAD FACTOR *integer* PERCENT ;
ON AREA[S] *area-name-1*[, … *area-name-n*]

4.8 CHANGE ORDER Directive

The "ORDER IS" clause in the set section is used to store members in a particular sequence. While SORTED, FIRST and PRIOR orders may be well suited for various types of production processing, these ORDER clauses may cause considerable overhead when used for a database reload. A common strategy on database reorganizations is to compile (@DDL) a special load schema that has the set order changed to "ORDER IS NEXT". In addition, member records are pre-ordered in the required sequence and stored using this special load schema. The CHANGE ORDER directive allows the set order to be changed to NEXT for database reloads, thus minimizing the overhead associated with set ordering and eliminating the need to compile a special load schema.

Format:

CHANGE ORDER TO NEXT ON SET *set-name-1*[, … *set-name-n*]

You may not change an "ORDER IS LAST" set to "ORDER IS NEXT" or "ORDER IS FIRST" unless PRIOR pointers are present. Without PRIOR pointers, an "ORDER IS LAST" set has an additional pointer on the owner (a PRIOR pointer) that is not present on owners of other "ORDER IS" types.

4.9 CHANGE TIP FILE CODE Directive

The CHANGE TIP directive will change the TIP file code of the current schema (named in the USE SCHEMA) as well as the specified subschema. The host language of the subschema used here may be either ASCII COBOL or DMU.

The CHANGE TIP FILE CODE directive is used to alter the TIP file code in the object schema and subschema. This alteration enables I-QU PLUS-1 to use a schema and subschema from an alternate TIP file without affecting the schema in the production TIP file. Using this directive in combination with other SCHUTL directives, allows the user to produce load

schemas for efficiency. The absolute schema and subschema must be copied to the specified alternate TIP file prior to being INVOKED by the I-QU PLUS-1 Processor.

A DMU subschema may also be modified using this directive, allowing the use of a single-thread version of the DMU to verify a database reorganization without having to put the new schema into production (multi-thread).

Format:

CHANGE TIP FILE CODE *subschema-name new-TIP-file-code*

4.10 IGNORE RESULT PROCESSING Directive

DMS provides for two types of RESULT processing. The first type of RESULT processing calculates a data item from other data values found on the record. This form of RESULT processing provides a "cross-footing" capability. It does not create any problems for database reorganization. The second type of RESULT processing, calculates a data item from data values that exist in a specified set membership. As members are added to the set, a RESULT data item in the owner is calculated. This process can be very resource intensive during database reorganization, especially when the values, which comprise the RESULT calculation, do not change.

This directive allows the user to disable the recalculation of RESULT items that are derived from member records. Care should be taken when changing items that are used in the derived calculation. In this case, disabling RESULT processing would require the RESULT item to be calculated programmatically.

Format:

IGNORE RESULT PROCESSING ON RECORD[S] *record-name-1*[, … *record-name-n*]

4.11 IGNORE SET Directive

The IGNORE SET directive modifies the specified record's description table, causing the DMR to ignore the record's participation in the specified set or sets. The pointer positions reserved for the specified sets will be present and may be modified using the I-QU PLUS-1 PUTPTR command without affecting the DMR. Sets named in the IGNORE SET directive must be sets in which the specified record participates as a member.

The temporary schema modified by the IGNORE SET directive must only be used for an I-QU PLUS-1 reload operation. All PFIX and verification programs must use the actual, unmodified schema.

The IGNORE SET can be used to save a great amount of I/O overhead when using I-QU PLUS-1 to reload records, which are members of sets whose owners reside in other areas. However, IGNORE SET may not be used on primary sets where a record's location mode is VIA SET.

Format:

IGNORE SET *record-name* SET[S] *set-name-1*[, … *set-name-n*]

4.12 USE COS ON SET Directive

The use of Location Mode of Owner (LMO) clause reduces the amount of programming that must be done by the application at the expense of additional processing by the DMR. When storing a member that is LMO, the DMR must first establish an owner higher in the structure that is an entry point record or a via set record whose primary set is selected based on currency (COS). Next, the sets are accessed downward in the structure ulilizing set order and in some cases searching for matching key criteria. The Current of Set (COS) facility

provides a storage mechanism with less overhead that is well suited for database reload. This directive allows set occurrence entry of LMO to be changed to COS for database reload efficiency.

Format:

USE COS ON SET[S] *set-name-1*[, … *set-name-n*]

4.13 SCHUTL Example Run

The following is an example of how the SCHUTL Processor would be used in a batch run to set up a schema and subschema for a reorganization using I-QU PLUS-1.

```
@RUN,D DBA01,,IQU
@COPY,A DMS*SCHEMAFILE.,TEMPSCHEMA.      <- Temp. schema object file.
@SCHUTL
USE SCHEMA PRD-SCHEMA FILE TEMPSCHEMA
CHANGE AREA LOOKS TO NOLOOKS
CHANGE AREAS TO NON-PREINIT
IGNORE SET ORDER-HEADER-REC SETS CUST-ORD, ORDER-OPEN
CHANGE TIP FILE CODE TRV-SUB 153
@EOF
```

At the completion of this run, the I-QU PLUS-1 user would copy the schema and subschema to TIP file 153. The modified schema and subschema may then be invoked from the alternate TIP schema file.

Chapter 5: PBLD and PFIX - Pointer Correction Utilities

To understand the purpose of the PBLD and PFIX utilities better, a brief review of database pointers, and how they are affected by reorganization, is useful. Each record in a database has a unique "address" composed of the area code, page number and page slot number. When an area is reorganized, records will probably be placed in different locations, changing the record address. Because I-QU PLUS-1 allows the reorganization of portions of a data structure (as opposed to reorganizing the entire data structure), pointers in non-reorganized areas that point into the reorganized area will have the old (and now incorrect) record address. To correct these database pointers, I-QU PLUS-1 allows the user to save the original address (DBP) of each record at unload time. Then, when the record is stored into its new location during the reload, its new DBP is also saved. An I-QU PLUS-1 command, XREF, is then executed to create a cross-reference record, which contains the record's original DBP and its new DBP.

During the reload phase of the reorganization, each record is cross-referenced so that when the reload is finished there will be a complete cross-reference of old database pointers to new database pointers, or stated differently, a complete list of the pre-reorganization addresses and the corresponding post-reorganization addresses. What is needed at this point is a way to find all the old addresses and replace them with the new addresses. When this is done, all records will be logically linked just as they were before the reorganization, even though they physically reside in different locations. The process of locating and replacing database pointers is one purpose of PBLD and PFIX. PBLD formats and optimizes the file created by the XREF command, and PFIX actually locates the old database pointers and replaces them.

5.1 PBLD - Pointer Cross-Reference Build Program

The purpose of PBLD is to build a database pointer cross-reference file to be used in a PFIX RELINK run. Input to this program is the database pointer (or database key) file created by the XREF command during an I-QU PLUS-1 reload. The file will contain the "old" (before reorganization DBPs) and "new" (after reorganization DBPs) database pointers for each record in the reorganized areas. If the area or areas were reloaded using several runs, creating more than one input pointer file, all pointer cross-reference files must be combined into a single cross-reference file by PBLD.

PBLD will sort the input file(s) by "old" database pointer (DBP), and block the output for efficiency. Two output files will be produced by PBLD. The DBPPARMS file will contain parameters describing attributes of affected areas: i.e., information needed by the PFIX program. The second file, DBPXREF, is the database pointer cross-reference file. If these files are not pre-assigned by the user, new cycles of both output files will be automatically catalogued during PBLD processing. Sort work files (XA through XZ) should be assigned in the PBLD run for optimum sort performance. If sort work files are not pre-assigned, PBLD

will automatically assign temporary files XA, XB and XC to fixed mass-storage with a minimum 500 tracks and a maximum of 5,000 tracks each.

The DBPXREF file (a PCIOS sequential file) contains cross-reference records. These records are two words long and consist of the old DBP and new DBP. PBLD detects duplicate old DBPs. This condition is caused by invalid data in the input XREF file. When a duplicate condition is encountered, PBLD will issue a message along with a dump of the first DBP record written to the DBPXREF file and the duplicate record that was to be written to the DBPXREF file. The database administrator should examine the DBPs to determine the area and page location. This information will help to determine the cause of the duplicate DBPs. If the "C" option is not present on the PBLD processor call, the PBLD program will terminate. Should the database administrator desire further information, the "C" option will allow the PBLD processor to continue and identify any other duplicate conditions.

5.1.1 PBLD Processor Call and Options

PBLD is executed as follows:

@PBLD,*options*

The PBLD processor name used will depend on the installation mode used when I-QU PLUS-1 was installed. The default is PBLD. See the section in the I-QU PLUS-1 Installation Guide called "The Product Files" for more information on the naming conventions used for processors. The options are:

- B Causes the execution of PBLD to be treated as a batch mode execution.
- D Causes the execution of PBLD to be treated as a demand mode execution. This option is the inverse of the B-Option.
- E Edit parameters only. No other action taken.
- L List internal parameter information.
- T is for KMSYS Worldwide debugging only. Use only if directed to by KMSYS Worldwide' personnel.

5.1.2 PBLD's Input Directives

PBLD requires three input directives: one to specify the schema to be used; one to describe each reorganized area; and one to indicate which database pointer files will be used as input. All parameters must start in position one.

5.1.2.1 USE SCHEMA Directive

The USE SCHEMA directive specifies the schema name and file of the old schema (or unload schema).

Format:

USE SCHEMA *schema-name* FILE *file-name*

5.1.2.2 AREAS Directive

The AREAS directive will specify the area(s) for which the pointer cross-reference file is being built; in other words, the area(s) being reorganized.

Format:

AREA[S] *area-name-1*[, … *area-name*-n]

5.1.2.3 DBPFILE Directive

The DBPFILE directive is used to specify a list of database pointer files for input to PBLD. Up to 50 database pointer files may be included.

Format:

DBPFILE[S] *file-name-1*[, ... *file-name-50*]

5.1.3 PBLD -Example

```
@ .  *** PRE-ASSIGN SORT WORK AND OUTPUT DBPXREF FILE ***
@ASG,T XA.,F70M/1000//1000,REM800
@ASG,T XB.,F70M/1000//1000,REM801
@ASG,T XC.,F70M/1000//1000,REM802
@ASG,UP DBPXREF.,F70M/500//1000,REM900
@.
@PBLD
USE SCHEMA PROD-SCH FILE DMS*SCHEMA
AREA CUST-MAST
DBPFILE REORG*DBPFILE.
@EOF
```

PBLD will produce an output that looks similar to this:

```
@PBLD PBLD IQU11R6 (Release 11R6) (991106 1752:13) 1999 Nov 18 Thu 1920:02
(C) Copyright 1983-1996 by KMSYS Worldwide, Inc.  All Rights reserved.
This program licensed for use by KMSYS Worldwide, INC.
   1             USE SCHEMA PROD-SCH FILE DMS*SCHEMA.
CSF: @ASG,A DMS*SCHEMA
CSF: @USE I$QUSCH.,DMS*SCHEMA
<<< Schema reference tables loaded >>>
   2             AREA CUST-MAST
   3             DBPFILE REORG*DBPFILE.
<<< End of PBLD parameters >>>
CSF:@ASG,UP DBPPARMS(+1).
CSF:@USE DBPPARMS.,DBPPARMS(+1).
*** XA WORK pre-assigned.
*** XB WORK pre-assigned.
*** XC WORK pre-assigned.
CSF:@ASG,A REORG*DBPFILE
CSF:@USE DBPINPUT,REORG*DBPFILE
*** DBPXREF pre-assigned.
**** DBPXREF FILE BUILT *****
DBPCOUNT = 000000500
DBPCOUNT = 000000500
Processed 0000000500 DBPS for area CUST-MAST Code:00003
CSF:@FREE DBPPARMS.
CSF:@FREE DBPXREF.
<<<< End of PBLD processing >>>>
```

5.2 PFIX - Pointer Fix Program

PFIX is a multi-purpose program with three distinctly different functions, each of which requires a different set of directives. The following subsections describe these functions and associated directives.

In all cases where non-TIP database areas are to be affected, they must be assigned to the PFIX run. Access to TIP areas by PFIX uses TIP I/O; therefore, TIP areas must be assigned to TIP.

5.2.1 PFIX Processor Call and Options

The PFIX Processor is executed as follows:

@PFIX,*options*

The PFIX processor name used will depend on the installation mode used when I-QU PLUS-1 was installed. The default is PFIX. See the section in the I-QU PLUS-1 Installation Guide called "The Product Files" for more information on the naming conventions used for processors. The options are:

- B Causes the execution of PFIX to be treated as a batch mode execution.
- C Continue processing, if possible, after error detection.
- D Cause the execution of PFIX to be treated as a demand mode execution. This option is the converse of the B-Option.
- E Edit parameters only. No area updates will occur.
- F Generate and verify page checksums as implemented under DMS 2200, Level 8R3 and earier.
- L List internal parameters.
- N Specifies no cross-reference check. Do not error if DBP cannot cross-reference.
- Q Quiet console messages.
- T For KMSYS Worldwide debugging only. Use only if directed to by KMSYS Worldwide personnel.
- U Generate and verify page checksums as implemented under the Universal Data System (UDS).
- V Disable the ability to do II-keyins.
- W Do not display warning messages.
- X For KMSYS Worldwide debugging only. Use only if directed to by KMSYS Worldwide personnel.

5.2.2 RELINK Set Relationships

RELINK is used to reestablish set relationships broken by the unloading and reloading of an area independent of other logically linked areas. The RELINK process uses the cross-reference file and the parameter file built by PBLD.

The descriptions of the areas to which records will be relinked are passed from PBLD via file DBPPARMS. The current DBPPARMS file under the default qualifier will automatically be assigned by PFIX, if one is not assigned prior to PFIX execution. The current DBPXREF, created by PBLD, will be handled in the same manner.

A temporary work file named TEMP will be required for the RELINK. If it is not pre-assigned, PFIX will assign it with a minimum of 500 tracks and a maximum of 5,000 tracks on fixed mass-storage. Sort work files (XA through XZ) should be assigned in the PFIX relink run for optimum sort performance. If sort work files are not pre-assigned, PFIX will automatically assign temporary file XA, XB and XC to fixed mass-storage with a minimum 500 tracks and a maximum of 5,000 tracks each. The actual required sizes of the TEMP and sort work files will vary greatly depending on the number of pointers that have to be altered.

The DBPXREF file (a PCIOS sequential file) contains cross-reference records. These records are two words long and consist of the old DBP and new DBP. PFIX matches the old DBP against the set pointers named on the RECORD directive of PFIX. If PFIX is unable to match a set pointer (i.e., PFIX cannot find a match in the DBPXREF file), PFIX will terminate with the following error message:

```
Error.  Sort DBP < old DBP
<ERR> Unable to cross-reference pointer.
```

If this occurs, PFIX will also display the "old DBP" from the DBPXREF file, a page reference where the set pointer ("Sort DBP") can be found (page number and word offset to the record on the page) and the actual set pointer.

Several conditions can cause this error. PFIX will attempt to interpret the condition in order to better identify the problem. For example, if the DBPXREF file is empty, PFIX will display an appropriate message. Other conditions are not as easily identified. Three of the more common conditions are malformed old DBPs, use of the wrong schema in PBLD/PFIX and misplaced set pointers (the GETPTR command in the unload program and the PUTPTR command in the reload program). A DBP can be malformed by moving an I-QU PLUS-1 variable to/from an RDA reference with the wrong data type in either the unload or reload program. The data type should always be UB9.

To investigate the problem further, use the DMU PRINT command to view the offending page and I-QU PLUS-1 processor to view the DBPXREF file on-line.

5.2.2.1 RELINK Directive

The RELINK directive specifies that PFIX is to perform a RELINK operation, and the schema to be used. The schema specified must be the schema currently in effect; i.e., the new schema after a reload. This directive must appear first.

Format:

RELINK USING *schema-name* FILE *file-name*

5.2.2.2 SEARCH Directive

This directive specifies in which areas to search for records to be relinked. Additionally the page range within the search areas may be limited, thus allowing more than one PFIX run to operate on the same area simultaneously. If simultaneous relinks are to be performed on an area, make sure the page ranges in each run do not overlap.

Format:

SEARCH AREA[S] *area-spec-1*[, … *area-spec-n*]

Where *area-spec-1* through *area-spec-n* has the format:

area-name[,*start-page*,*end-page*]

5.2.2.3 RECORD Directive

This parameter indicates which records, and within them, which set pointers will be examined and possibly altered during the area search. Any record in the area not specified by a RECORD parameter will be ignored by PFIX. PFIX will find and examine each pointer specified for each set specified to determine if it needs to be cross-referenced and possibly replaced.

Format:

RECORD record-name SETS *set-name-1*[/*opn*][, … *set-name-n*[/*opn*]]

The optional /*opn* (owner/prior/next) specification determines which pointers in the record will be fixed or ignored during the PFIX process. The /*opn* specification may be a value 0

(zero is the default) to 111. The units position controls processing of the NEXT pointer of a set. The tens position represents the PRIOR pointer and controls processing of the prior pointer of a set. Similarly, the hundreds position controls the processing of the OWNER pointer in a set.

A zero (0) in any position of the /*opn* specification implies that the pointer is to be fixed; a one (1) implies, do not fix. Note: Leading zeroes are not required.

For example, the following PFIX directive would ignore fixing the next pointer for the ORDER-OPEN set in the ORDER-CONTROL-REC record:

```
RECORD ORDER-CONTROL-REC SET ORDER-OPEN/1
```

This directive is equivalent to:

```
RECORD ORDER-CONTROL-REC SET ORDER-OPEN/01
```

or

```
RECORD ORDER-CONTROL-REC SET ORDER-OPEN/001
```

To fix only the owner pointer (ignoring the prior and next pointers), the directive would appear as follows:

```
RECORD ORDER-CONTROL-REC SET ORDER-OPEN/11
```

An equivalent directive would be:

```
RECORD ORDER-CONTROL-REC SET ORDER-OPEN/011
```

To fix the prior and next pointers yet leave the owner pointer unaltered, the directive would be coded in the following manner:

```
RECORD ORDER-CONTROL-REC SET ORDER-OPEN/100
```

Fixing only the prior pointer, would require the following notation:

```
RECORD ORDER-CONTROL-REC SET ORDER-OPEN/101
```

5.2.3 DELINK Set Ownership

DELINK can be used when an area containing member record types, whose owners occur in a different area, is to be initialized and reloaded. The DELINK operation is always performed on OWNER record types. The owner record is DELINKED from the set chain it owns. Each set occurrence is made a null, or empty set. Using DELINK avoids the overhead of using DML to delete all member records from the set occurrence. After the DELINK is done, the member record area can be initialized and reloaded. The RELINK of member records into the proper set occurrence is accomplished using normal DML commands (i.e., making the proper owner record current before the member is stored).

5.2.3.1 DELINK Directive

The DELINK directive specifies that PFIX is to perform a DELINK operation, and the schema to be used. The schema specified must be the schema currently in effect. This directive must appear first.

Format:

DELINK USING *schema-name* FILE *file-name*

5.2.3.2 SEARCH Directive

This directive specifies in which areas to search for records to be delinked. Additionally the page range within the search areas may be limited, thus allowing more than one PFIX run to operate on the same area simultaneously. If simultaneous delinks are to be performed on an area, make sure the page ranges in each run do not overlap.

Format:

SEARCH AREA[S] *area-spec-1*[, ... *area-spec-n*]

Where *area-spec-1* through *area-spec-n* has the format:

area-name[,*start-page*,*end-page*]

5.2.3.3 RECORD Directive

This parameter indicates which records, and within them, which set ownership pointers will be set to null (empty) sets during the area search. Any record in the area not specified by a RECORD parameter will be ignored by PFIX.

Format:

RECORD record-name SETS *set-name-1*[, ... *set-name-n*]

5.2.4 DBPCHANGE Global Pointer Change

This function of PFIX is the ability to change database area codes and/or to modify the format of database pointers (and database keys contained in pointer array records) without performing a database reorganization. Database pages are read directly from mass storage, pointers and array record database keys are modified, and the page is rewritten to mass storage. A PFIX DBPCHANGE run incurs two I/O's (read and write) for each page scanned. A DBPCHANGE run requires no internal sort operations.

The DBPCHANGE function of PFIX requires information from the current object schema and the old object schema. Area information is compared between the new and old schema to determine the affect of the change. The comparison will match areas by name; therefore, area names must not change between schemas. The only attributes that may change between old and new schemas are:

- Area codes;
- Area code bits changed due to change in AREA CONTROL clause in schema;
- Page bits changed due to change in ALLOCATE and/or EXPANDABLE clauses.

Any change must be reflected in the schema for it to be recognized by PFIX. For example, if data was copied from one area to another area of the same size within the same schema, PFIX could not recognize the change. Even though the area codes in the receiving area would be incorrect (the same as the original area), the schema definition has not changed and PFIX is driven from the schemas.

5.2.4.1 DBPCHANGE Directive

This directive specifies that PFIX is to perform a DBPCHANGE operation, and both the old and new schemas that are to be used. The first schema specified must be the schema currently in effect. This directive must appear first.

Format:

DBPCHANGE USING *new-schema* FILE *file-name* ;
OLDSCHEMA *old-schema-name* FILE *file-name*

5.2.4.2 AREAS Directive

The AREAS directive is used to limit the scope of the DBPCHANGE operation. If omitted, every area that appears in both schemas will be scanned. This parameter is normally included when the user is sure which areas are affected and which are not.

Format:

AREA[S] *area-name-1*[, ... *area-name-n*]

If the AREAS parameter is omitted, ALL areas will be assumed.

5.2.5 DMS 2200 Page Checksums

While DMS 2200 page checksums were not officially implemented until DMS 2200 Level 9R1, many user sites locally modified their DMRs to use the checksum code which was already in DMS 2200 (element IO05).

Since DMS 2200, operating under UDS, implemented the checksum feature in a way incompatible with the DMS level 8 local code, PFIX now has two additional options, F and U. If neither option is specified, the checksum word on the page is not examined or modified. If the F-option is set on the PFIX processor call, the handling of checksums is identical to the handling in level 8 of DMS. If the U-option is set, checksums will be generated and verified in the same manner as is done in UDS with the checksum feature set to ON for all areas.

5.2.6 No Cross-reference Check Considerations

If the no cross-reference check execute option (N) is used, PFIX will ignore any pointer for which no old pointer is found in the pointer cross-reference file. This option must only be used when relinking sets where it is anticipated that not all member records being scanned will have pointers affected by the movement of other records. For example, if the owners of a set were unloaded and reloaded, and the member records reside in the same area, only the set pointers of member records that point to the relocated owner records (last and/or first members of sets) will have been cross-referenced. Member-to-member pointers would have remained intact because member records have not been moved. If this parameter is not specified, and a pointer is found in a record being relinked that has no corresponding entry in the cross-reference file, a fatal PFIX runtime error will result. A fatal PFIX runtime error results because PFIX cannot distinguish between a member-to-owner pointer and a member-to-member pointer.

5.2.7 PFIX Runtime Status

Several informative messages will be displayed on the system console by a PFIX run. These messages will keep the user informed as to the process of the run. One message of particular interest is the following:

```
**** AREA xxxxxxxxxxx has been updated, recovery required before restart ****
```

Where xxxxxxxxxx will be an area name.

This message tells the user when an area has actually been modified. If for some reason the PFIX run does not complete, areas for which this message has not been displayed will not have to be recovered before restart.

Please note, once this message has been displayed for an area, DO NOT ATTEMPT A RESTART WITHOUT FIRST RECOVERING THE AREA.

5.2.8 PFIX II Key-in

In addition to the automatically displayed messages, the user may obtain current status information from PFIX while it is running via the I I key-in at the system console. The format of the II key-in is:

II run-id {P | S}

The run-id will be the run name under which PFIX is executing.

If the P-option is used, PFIX will display the name of the current area being scanned, the current page number, the total number of DBPs replaced, and the total number of pages modified. If the S-option is used, the above information will be preceded by the number of records encountered for each record type specified, and the number of pointers checked up to this point. This information will automatically be printed at the end of the run.

5.2.9 Examples of PFIX Runs

The following examples of DELINK, RELINK and DBPCHANGE runs are provided to help the user in recognizing normal execution of various PFIX run functions:

To execute a DELINK Run:

```
@PFIX
DELINK USING PROD-SCHEMA FILE DMS*SCHEMA
SEARCH AREA CUST-MAST
RECORD CUST-KEY-REC SETS CUST-ADDR, CUST-ORD
@EOF
```

The actual output of the above run would look similar to the following:

```
@PFIX
PFIX IQU11R6 (Release 11R6) (991106 1753:13) 1999 Nov 18 Thu 1920:08
(C) Copyright 1983-1996 by KMSYS Worldwide, Inc.  All Rights reserved.
This program licensed for use by KM SYSTEMS, INC.
RLKID=$96319
   1             DELINK USING PROD-SCHEMA FILE DMS*SCHEMA
CSF: @ASG,A DMS*SCHEMA
CSF: @USE I$QUSCH.,DMS*SCHEMA
<<< Schema reference tables loaded >>>

   2             SEARCH AREA CUST-MAST
   3             RECORD CUST-KEY-REC SETS CUST-ADDR, CUST-ORD
<<< End of user parameters >>>>

**** Starting DELINK
**** Areas are EXEC, assumed to be assigned to this run.
Starting scan of CUST-MAST no.  pages = 00050 page size = 0000000896
**** AREA CUST-MAST has been updated, recovery required before restart
****
**** End scan of area CUST-MAST

**** Final DELINK statistics:
Record:CUST-KEY-REC
Found 0000000500 recs., checked 0000001000 ptrs.
**** End-of-run DELINK ****
Current DELINK statistics:
Record:CUST-KEY-REC
Found 0000000500 recs., checked 0000001000 ptrs.
Current area              at page 000050
Total DBPS replaced = 0000001000
Total pages modified = 0000000043
End of DELINK statistics
*** Total DBPS actually replaced = 0000001000
*** Total pages modified = 0000000043
```

Many of the status messages shown above will also appear on the system console during run execution.

To execute a RELINK run:

```
@ .  *** PRE-ASSIGN SORT WORK AND OUTPUT DBPXREF FILE ***
@ASG,T XA.,F70M/1000//1000,REM800
@ASG,T XB.,F70M/1000//1000,REM801
@ASG,T XC.,F70M/1000//1000,REM802
@ASG,T TEMP.,F70M/1000//1000,REM900
@.
@PFIX
```

```
RELINK USING PROD-SCH FILE DMS*SCHEMA.
SEARCH AREAS ORDERS, CUSTADDR
RECORD CUST-ADDR-REC SET CUST-ADDR
RECORD ORDER-HEADER-REC SET CUST-ORD
@EOF
```

The above PFIX RELINK run would produce the following output:

```
@PFIX
PFIX IQU11R6 (Release 11R6) (991106 1753:13) 1999 Nov 18 Thu 1920:08

(C) Copyright 1983-1996 by KMSYS Worldwide, Inc.  All Rights reserved.
This program licensed for use by KMSYS Worldwide, INC.
RLKID=$96319

   1             RELINK USING PROD-SCH FILE DMS*SCHEMA.
CSF: @ASG,A DMS*SCHEMA
CSF: @USE I$QUSCH.,DMS*SCHEMA
<<< Schema reference tables loaded >>>
   2             SEARCH AREAS ORDERS, CUSTADDR
   3             RECORD CUST-ADDR-REC SET CUST-ADDR
   4             RECORD ORDER-HEADER-REC SET CUST-ORD
<<< End of user parameters >>>>
CSF:@ASG,A DBPPARMS.
CSF:@ASG,A DBPXREF.
*** TEMP WORK pre-assigned.
*** XA WORK pre-assigned.
*** XB WORK pre-assigned.
*** XC WORK pre-assigned.
DBP Params: area-code,area-bits,rec-bits,slot-bits
00003,10,17,09

**** Starting RELINK
**** Areas are EXEC, assumed to be assigned to this run.

Starting scan of ORDERS no.  pages = 00500 page size = 0000000896
**** End scan of area ORDERS
**** Entering DBP match with 0000000500 records.
**** AREA ORDERS has been updated, recovery required before restart ****

Starting scan of CUSTADDR no.  pages = 00050 page size = 0000000896
**** End scan of area CUSTADDR
**** Entering DBP match with 0000000500 records.
**** AREA CUSTADDR has been updated, recovery required before restart ****

**** Final RELINK statistics:
Record:CUST-ADDR-REC
Found 0000000500 recs., checked 0000000500 ptrs.
Record:ORDER-HEADER-REC
Found 0000001204 recs., checked 0000000500 ptrs.
**** End-of-run RELINK ****
Current RELINK statistics:
Record:CUST-ADDR-REC
Found 0000000500 recs., checked 0000000500 ptrs.
Record:ORDER-HEADER-REC
Found 0000001204 recs., checked 0000000500 ptrs.
Current area              at page 000050
Total DBPS replaced = 0000000998
```

```
Total pages modified = 0000000294
*** Extract and sort -phase 1 ****
End of RELINK statistics
*** Total DBPS actually replaced = 0000000998
*** Total pages modified = 0000000294
```

To execute a DBPCHANGE run:

```
@SYS$LIB$*IQU.PFIX,L
DBPCHANGE USING PROD-SCH FILE DMS*SCHEMA ;
 OLDSCHEMA PROD-SCH FILE DMS*OLDSCHEMA
@EOF
```

In this case, the AREA CONTROL clause of the schema was changed from 128 areas to 600 areas, and the EXPANDABLE clause for the CUST-MAST area was changed from 99,999 pages to 5,000 pages. These changes are reflected in the area comparison portion of the DBPCHANGE output listing shown below:

```
@PFIX
PFIX IQU11R6 (Release 11R6) (991106 1753:13) 1999 Nov 18 Thu 1920:08
(C) Copyright 1983-1996 by KMSYS Worldwide, Inc.  All Rights reserved.
This program licensed for use by KMSYS Worldwide, INC.
RLKID=$96319
   1             DBPCHANGE USING PROD-SCH FILE DMS*SCHEMA ;
   2             OLDSCHEMA PROD-SCH FILE DMS*OLDSCHEMA
CSF: @ASG,A DMS*SCHEMA
CSF: @USE I$QUSCH.,DMS*SCHEMA
<<< Schema reference tables loaded >>>
CSF: @ASG,A DMS*OLDSCHEMA
CSF: @USE I$QUSCH.,DMS*OLDSCHEMA
<<< Old schema reference tables loaded >>>

*** Area comparison follows:

_____________________________________________________________________________
_____________________________________________________________________________
Area: CUSTADDR changed:
 Old area code & DBP bit format = 00001 10/17/09
 New area code & DBP bit format = 00010 08/17/11
_____________________________________________________________________________
Area: CUST-MAST changed:
 Old area code & DBP bit format = 00003 10/17/09
 New area code & DBP bit format = 00012 08/13/15
_____________________________________________________________________________
Area: CUSTADDRX changed:
 Old area code & DBP bit format = 00002 10/17/09
 New area code & DBP bit format = 00011 08/17/11
_____________________________________________________________________________
Area: ORDERS changed:
 Old area code & DBP bit format = 00004 10/17/09
 New area code & DBP bit format = 00013 08/17/11
_____________________________________________________________________________
CSF: @ASG,A DMS*SCHEMA
CSF: @USE I$QUSCH.,DMS*SCHEMA
<<< New schema reference tables re-loaded >>>
<<< End of user parameters >>>>

**** Starting DBPCHG
**** Areas are EXEC, assumed to be assigned to this run.
```

```
Starting scan of CUSTADDR     no.  pages = 00050 page size = 0000000896
*** Data area
**** AREA CUSTADDR     has been updated, recovery required before restart
****
**** End scan of area CUSTADDR
Current DBPCHG statistics:
Record:CUST-ADDR-REC
Found 0000000500 recs., checked 0000001000 ptrs.
Current area CUSTADDR     at page 000050
Total DBPS replaced = 0000000500
Total pages modified = 0000000050
End of DBPCHG statistics

Starting scan of CUST-MAST    no.  pages = 00050 page size = 0000000896
*** Data area
**** AREA CUST-MAST    has been updated, recovery required before restart
****
**** End scan of area CUST-MAST
Current DBPCHG statistics:
Record:CUST-KEY-REC
Found 0000000500 recs., checked 0000002000 ptrs.
Current area CUST-MAST    at page 000050
Total DBPS replaced = 0000002043
Total pages modified = 0000000100
End of DBPCHG statistics

Starting scan of CUSTADDRX    no.  pages = 00005 page size = 0000000896
*** I.S.  index area
**** AREA CUSTADDRX    has been updated, recovery required before restart
****
**** End scan of area CUSTADDRX
Current DBPCHG statistics:
Current area CUSTADDRX    at page 000005
Total DBPS replaced = 0000002044
Total pages modified = 0000000105
End of DBPCHG statistics

Starting scan of ORDERS       no.  pages = 00500 page size = 0000000896
*** Data area
**** AREA ORDERS       has been updated, recovery required before restart
****
**** End scan of area ORDERS
Current DBPCHG statistics:
Record:ORDER-HEADER-REC
Found 0000001204 recs., checked 0000007224 ptrs.
Record:ORDER-LINE-REC
Found 0000015575 recs., checked 0000046725 ptrs.
Current area ORDERS       at page 000500
Total DBPS replaced = 0000039643
Total pages modified = 0000000605
End of DBPCHG statistics

Total DBPS replaced = 0000039643
Total pages modified = 0000000605
End of DBPCHG statistics
*** Total DBPS actually replaced = 0000039643
```

```
*** Total pages modified = 0000000605
```

Index

If you would like to help us make our documentation better, please take a few moments to complete this form and return it to KMSYS Worldwide. We are always looking for ways to improve our products and your feedback will help us reach our goal.

Name _____

Company _____

Address _____

_____

City _____ State/Province _____

Country _____ Zip/Mail Code _____

Document Name _____ OS Level _____

KMSYS Worldwide Product _____ Level _____

Please rate the documentation on a scale of 1 to 5:

	5	4	3	2	1	
Complete	❑	❑	❑	❑	❑	Incomplete
Accurate	❑	❑	❑	❑	❑	Inaccurate
Usable	❑	❑	❑	❑	❑	Unusable
Readable	❑	❑	❑	❑	❑	Unreadable
Understandable	❑	❑	❑	❑	❑	Unintelligible
Attractive	❑	❑	❑	❑	❑	Unattractive
Excellent	❑	❑	❑	❑	❑	Poor

What information did you expect to find that was omitted?

_____

Is more information needed? ❑ Yes ❑ No. If yes, on what topic?

_____

Did you find factual errors in the documentation? ❑ Yes ❑ No. If yes, please give page number and description of the error.

_____

If the documentation is difficult to understand, please specify page number and problem.

_____

Is the documentation intimidating? ❑ Yes ❑ No.

_____

Are the manuals: ❑ Too long? ❑ Too short? ❑ About the right length?

_____

Other suggestions or comments? (Use back of form if necessary.)

_____

(Additional Comments)

. . . . . Fold along dotted line. . . . . .

KMSYS
WORLDWIDE, INC
P.O. Box 669695
Marietta, GA 30066
U.S.A.

Attn: Technical Documentation Section