IBM IMS DataPropagator for z/OS

Reference

Version 3 Release 1

SC27-1210-02

IBM IMS DataPropagator for z/OS

Reference

Version 3 Release 1

SC27-1210-02

Note:
Before using this information and the product it supports, be sure to read the general information under “Notices” on page 777.

Third Edition (May 2006)

This edition applies to Version 3 Release 1 of IBM IMS DataPropagator for z/OS, 5655-E52, and to any subsequent releases until otherwise indicated in new editions.

This edition replaces SC27-1210-01.

Abstract

This online book supports the task of administering IMS™ DPROP by describing IMS DPROP utilities, control statements, and tables needed to extract and propagate data.

Contents

Figures

About This Information

This book is designed to help database administrators and system operators perform data propagation tasks.

This softcopy book is available only in PDF and BookManager® formats. This book is available on the z/OS® Software Products Collection Kit, SK3T-4270. You can also get the most current versions of the PDF, BookManager, and Information Center formats by going to the IBM® Data Management Tools Web site at www.ibm.com/software/data/db2imstools and linking to the Library page.

What is New in Version 3, Release 1

This edition, which is available in softcopy format only, includes technical and editorial changes.

IMS DataPropagator™ (IMS DPROP) Version 3, Release 1 presents improvements to both the product and the product library.

Product Changes

IMS DataPropagator Version 3 Release 1 provides a new, MQSeries-based asynchronous (MQ-ASYNC) propagation of IMS database changes to DB2® tables. With MQ-ASYNC enterprises can implement both:

- Near-Real-Time Propagation - With Near-Real-Time propagation, the delay between the update of the IMS database and the update of the DB2 tables can often be as short as one or two seconds.
- Point-In-Time Propagation - With Point-In-Time propagation, the data content of the DB2 tables matches the IMS database content at a previous clearly identified logical point in time. For example, an enterprise might decide that the content of the DB2 tables will match the following point in times: the logical end of a business day, the logical end of a business month, or the end of a specific IMS jobstream that updated the IMS databases.
- IMS-to-IMS propagation - With IMS-to-IMS propagation, in conjunction with the MQ-ASYNC method of propagation, you can propagate IMS databases where the IMS source and the IMS target are on different z/OS images.

Product Library Changes

The Version 3 Release 1 library has been updated to support the addition of MQSeries® asynchronous propagation.

There are now three *Administrators Guides*, one for each primary mode of propagation:

- *IMS DPROP Administrators Guide for MQSeries Asynchronous Propagation*
- *IMS DPROP Administrators Guide for Log Asynchronous Propagation*
- *IMS DPROP Administrators Guide for Synchronous Propagation*

There is also a new book, *IMS DataPropagator for z/OS: Concepts*, which provides a conceptual description of data propagation.

Terms Used in This Book

The following terms are synonymous in this book:

- *File* and *data set*
- *DXT™* and *DataRefresher™*

  Unless a specific version or release is referenced, these terms refer to either of the following products:
  - DXT Version 2 Release 5
  - DataRefresher Version 1 or higher
- Databases that have been *quiesced* or set to *READONLY status*.

  In all cases, these terms refer to either or both of the following:
  - Any propagatable database, except for DEDBs, that has been set to READONLY status.
  - DEDBs that have been taken offline with a /DBR command

References to DataRefresher and DXT in this book refer to only host activities. This book assumes that you will use batch and command statements, *not* the DataRefresher workstation component.

Selector and *Receiver* (capitalized) refer to the IMS DPROP Selector and Receiver features. However, *selector* and *receiver* (not capitalized) refer to user-created functions.

IMS DPROP books use the term “child” instead of the term “dependent.” For example, IMS DPROP books use the terms “child table” and “child rows” instead of DB2 terms “dependent table” and “dependent rows.” The term “child” is used so that terms for IMS and DB2 are similar.

What You Should Know

This book assumes you understand what data propagation is and the business reasons for propagating data. Information on these topics is in IMS DPROP An Introduction.

You can find additional conceptual information on data propagation in IMS DataPropagator for z/OS: Concepts.

This book also assumes you have a basic understanding of IMS, DB2, and DataRefresher concepts and functions.

What Is in This Book

This book describes IMS DPROP job control statements, IMS DPROP control statements and IMS DPROP utilities.

This book consists of the following parts:

- Part 1: Common JCL and control statements for IMS DPROP Components

  Chapter 1 describes JCL statements that are common to most IMS DPROP components and utilities.

  Chapter 2 describes the IMS DPROP control statements that are provided in the //EKYIN Input data set and that are common to most IMS DPROP components and utilities.
- Part 2: Defining Propagation Requests

This part describes how to define the IMS DPROP Propagation Requests (PRs). PRs describe to IMS DPROP which data should be propagated and how the data should be transformed (e.g. from hierarchical format to relational format) during the propagation process.

Chapter 3 describes how to define the PRs with DataRefresher commands.

Chapter 4 describes how to define the PRs in the MVG Input Tables, without use of DataRefresher.

- Part 3: IMS DPROP Directory Tables

  This part describes the IMS DPROP directory tables. They contain the PRs and the mapping information used by multiple IMS DPROP components such as the SCU, the RUP®, the CCU, the Receiver, and the Apply.

- Part 4: Utilities for Synchronous Propagation

  This part describes use of the SCU utility for synchronous propagation.

- Part 5: Utilities for LOG-ASYNC Propagation

  This part describes the use of various IMS DPROP components and utilities for LOG-ASYNC Propagation, such as:
  - The Status Change utility for LOG-ASYNC
  - The Selector Control File and the Selector Control File utilities
  - The LOG-ASYNC Selector
  - The LOG-ASYNC Receiver
  - PRDS Registration utility
  - CDCDS utilities

- Part 6: Components and Utilities for MQ-ASYNC Propagation

  This part describes the use of the three following MQ-ASYNC components and utilities.
  - Capture Component of MQ-ASYNC
  - Apply Component of MQ-ASYNC
  - Capture System utility

- Part 7: Consistency Check Utility (CCU)

  This part describes how the CCU is used to check the consistency between the IMS database content and the DB2 table content. It also describes how the CCU can help to reinstate consistency between IMS databases.

- Part 8: Audit Extract Utility

  This part describes how the Audit Extract utility can be used to extract SMF records that have been written by IMS DPROP and load these records into DB2 Audit Trail Tables for eventual SQL-based queries.

- Part 9: IMS-to-IMS propagation

  This part describes how to use IMS DataPropagator to propagate data changes from a source IMS database to a target IMS database copy of that source.

- Part 10: Appendixes

Service updates and support information

To find service updates and support information, including software fixpaks, PTFs, Frequently Asked Question (FAQs), technical notes, troubleshooting information, and downloads, refer to the following Web address: www.ibm.com/software/data/db2imstools/support.html

Where to find information

The Data Management Tools Library Web page provides current product documentation that you can view, print, and download. To locate publications with the most up-to-date information, refer to the following Web address: www.ibm.com/software/data/db2imstools/library.html

IBM Redbooks™ that cover Data Management Tools are available from the following Web address: www.ibm.com/software/data/db2imstools/support.html

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other Data Propagator documentation:

- Use the online reader comment form located at www.ibm.com/software/data/db2imstools/rcf/
- Send your comments by e-mail to comments@us.ibm.com. Be sure to include the name of the book, the part number of the book, the version of Data Propagator and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

How to Read the Syntax Diagrams

The following rules apply to the syntax diagrams used in this book:

Arrow symbols

Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

►►—— Indicates the beginning of a statement.

——► Indicates that the statement syntax is continued on the next line.

►—— Indicates that a statement is continued from the previous line.

——►◄ Indicates the end of a statement.

Diagrams of syntactical units other than complete statements start with the ►—— symbol and end with the ——► symbol.

Conventions

- Keywords, their allowable synonyms, and reserved parameters, appear in uppercase for MVS™ and OS/2® platforms, and lowercase for UNIX® platforms. These items must be entered exactly as shown.
- Variables appear in lowercase italics (for example, *column-name*). They represent user-defined parameters or suboptions.
- When entering commands, separate parameters and keywords by at least one blank if there is no intervening punctuation.
- Enter punctuation marks (slashes, commas, periods, parentheses, quotation marks, equal signs) and numbers exactly as given.
- Footnotes are shown by a number in parentheses, for example, (1).
- A » symbol indicates one blank position.

Required items

Required items appear on the horizontal line (the main path).

```
►►─REQUIRED_ITEM──────────────────────────────────────────────────►◄

                   ┌─optional_item─┐
►►─REQUIRED_ITEM───┴───────────────┴──────────────────────────────►◄
```

Optional Items

Optional items appear below the main path.

```
►►─REQUIRED_ITEM──┬──────────────┬──────────────────────────────────►◄
                  └─optional_item─┘
```

If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

Multiple required or optional items

If you can choose from two or more items, they appear vertically in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.

```
►►─REQUIRED_ITEM──┬─required_choice1─┬──────────────────────────────►◄
                  └─required_choice2─┘
```

If choosing one of the items is optional, the entire stack appears below the main path.

```
►►─REQUIRED_ITEM──┬──────────────────┬──────────────────────────────►◄
                  ├─optional_choice1─┤
                  └─optional_choice2─┘
```

Repeatable items

An arrow returning to the left above the main line indicates that an item can be repeated.

```
                   ┌──────────────┐
                   ▼              │
►►─REQUIRED_ITEM───┴─repeatable_item┴──────────────────────────────►◄
```

If the repeat arrow contains a comma, you must separate repeated items with a comma.

```
                   ┌─,────────────┐
                   ▼              │
►►─REQUIRED_ITEM───┴─repeatable_item┴──────────────────────────────►◄
```

A repeat arrow above a stack indicates that you can specify more than one of the choices in the stack.

Default keywords

IBM-supplied default keywords appear above the main path, and the remaining choices are shown below the main path. In the parameter list following the syntax diagram, the default choices are underlined.

```
►►──REQUIRED_ITEM──┬─default_choice──┬──────────────────────────────►◄
                   │                 │
                   ├─optional_choice─┤
                   └─optional_choice─┘
```

IMS-specific syntax information

Fragments

Sometimes a diagram must be split into fragments. The fragments are represented by a letter or fragment name, set off like this: `| A |`. The fragment follows the end of the main diagram. The following example shows the use of a fragment.

```
►►──STATEMENT──item 1──item 2──┤ A ├──────────────────────────►◄
```

A:

```
├──┬─item 3─┬──KEYWORD──┬──┬─item 5─┬──┬──────────────────────┤
   └─item 4─┘           │  └────────┘  │
                        └──────────────┘
                           └─item 6─┘
```

Substitution-block

Sometimes a set of several parameters is represented by a substitution-block such as **<A>**. For example, in the imaginary /VERB command you could enter `/VERB LINE 1`, `/VERB EITHER LINE 1`, or `/VERB OR LINE 1`.

```
►►──/VERB──┬──────┬──LINE──line#──────────────────────────►◄
           └─<A>──┘
```

where <A> is:

```
►►──┬─EITHER─┬──────────────────────────────────────────►◄
    └─OR─────┘
```

Parameter endings

Parameters with number values end with the symbol '#', parameters that are names end with 'name', and parameters that can be generic end with '*'.

```
►►──/MSVERIFY──┬─MSNAME──msname──┬──────────────────────────►◄
               └─SYSID──sysid#──┘
```

The MSNAME keyword in the example supports a name value and the SYSID keyword supports a number value.

Part 1. Common JCL and control statements for IMS DPROP Components

Chapter 1. Common JCL for IMS DPROP Components

Some JCL DD statements are used by the IMS DPROP components to perform common functions. Table 1 lists the common DD statements and where you are likely to use them. This topic describes the DD statements in detail including their syntax and examples.

Table 1. Common DD Statements

Statement	Where Used
//STEPLIB //JOBLIB	To specify the library containing: • IMS DPROP modules • DB2 modules • IMS modules
//EKYRESLIB	To allocate an APF library
//EKYSTATF	To describe the IMS DPROP status file used for synchronous propagation.
//EKYIN	To provide IMS DPROP control statements
//EKYLEOPT	To provide Language Environment/370 (LE/370) run time options
//EKYPRINT	To print messages or other output
//EKYTRACE	To write trace records for job steps
//EKYLOG	To define a data set to contain trace records
//EKYSNAP	To write SNAPs of the entire OS/VS task used to execute IMS DPROP
//EKYWTO	To write messages to the operator console or to a destination you specify

The following topics provide additional information:

- "//STEPLIB and //JOBLIB Statements "
- "//EKYRESLB DD Statement" on page 4
- "//DFSESL DD Statement " on page 4
- "//EKYSTATF DD Statement " on page 5
- "//EKYIN DD Statement " on page 5
- "//EKYLEOPT DD Statement " on page 6
- "//EKYPRINT DD Statement " on page 7
- "//EKYTRACE DD Statement " on page 7
- "//EKYLOG DD Statement " on page 8
- "//EKYSNAP DD Statement " on page 9
- "//EKYWTO DD Statement " on page 10

//STEPLIB and //JOBLIB Statements

If any IMS DPROP modules, SQL update modules, or exit routines are not in a link pack area or link-list library, then your job must use a //STEPLIB or //JOBLIB statement to identify the library where they are located. For example, you would need to specify:

- The load library containing the IMS DPROP modules
- The load library containing the DB2 modules, if you are executing the Receiver or the APPLY program
- The load library containing the IMS modules, if you are executing the Selector

- The load library containing the MQSeries modules, if you are executing the MQ-ASYNC Capture component, the MQ-ASYNC APPLY program, or the CUT utility.

Figure 1 shows an example of a //STEPLIB DD statement that identifies a shared data set named DPROP.EKYRESLIB. You would use the same syntax for a //JOBLIB statement.

```
//STEPLIB DD  DSN=DPROP.EKYRESLB,DISP=SHR
```

Figure 1. Example of //STEPLIB DD Statement

//EKYRESLB DD Statement

The IMS DPROP modules that execute in an MVS/ESA™ authorized mode must be loaded from an MVS authorized program facility (APF) library. If you do not have an exit routine that performs dynamic allocation, you can use a JCL DD statement with the EKYRESLB DD name to allocate the APF library, as shown in Figure 2.

Note that the IMS DPROP modules executing in authorized mode are also loaded from the library found in the //EKYRESLB DD statement. Therefore, the library you name must also be an APF library and should be protected through RACF® or a suitable equivalent.

```
//EKYRESLB DD   DSN=DPROP.EKYRESLB,DISP=SHR
```

Figure 2. Example of //EKYRESLB DD Statement

If you do not use the //EKYRESLB DD statement in your JCL, then IMS DPROP dynamically allocates the data set name you specified during DPROPGEN to the EKYRESLB DD name. IMS DPROP loads the modules that execute in authorized mode from that data set name.

//DFSESL DD Statement

For IMS online dependent regions that use external subsystems (such as DB2 or MQSeries), IMS might require a //DFSESL DD statement. This statement describes the APF-authorized load module libraries containing IMS modules and the external subsystem modules. An example of the //DFSESL DD statement follows:

```
//DFSESL     DD       DSN=IMS.RESLIB,DISP=SHR
//           DD       DSN=DSN.SDSNLOAD,DISP=SHR
//           DD       DSN=DSN.SDSNEXIT,DISP=SHR
//           DD       DSN=CSQ.MQSERIES.LOAD,DISP=SHR
```

Figure 3. Example of //DFSESL DD statement

//EKYSTATF DD Statement

The //EKYSTATF DD statement describes the IMS DPROP status file used for synchronous propagation. It is an OS/VS sequential data set, and its name must be specified as an IMS DPROP status file during IMS DPROP customization and DPROPGEN. The //EKYSTATF DD statement has the following data set characteristics, which are specified internally by IMS DPROP:

- RECFM=F or RECFM=FB
- LRECL=120

The data set described by the //EKYSTATF DD statement is:

- Required in the JCL of the IMS batch and dependent regions that perform synchronous propagation
- Required for most executions of the IMS DPROP SCU
- Optional for the execution of the IMS DPROP MVGU

Figure 4 is an example of an //EKYSTATF DD statement.

```
//EKYSTATF   DD   DSN=DPROP.TEST.STATUSF,DISP=SHR
```

Figure 4. Example of //EKYSTATF DD Statement

//EKYIN DD Statement

You use the //EKYIN DD statement to provide control statements to IMS DPROP. Chapter 2, “//EKYIN Data Set control statements ,” on page 11 provides more information on the //EKYIN control statements.

The //EKYIN DD statement must be either a sequential OS/VS data set or a member of an OS/VS partitioned data set. It must have RECFM=F or FB, and LRECL=80.

In synchronous propagation, job steps that invoke the RUP or HUP can use the following //EKYIN control statements:

PROP LOAD
PROP OFF
PROP SUSP
RESIDENT
TRACE
TRDEST

With MQ-ASYNC, jobsteps that update propagated IMS databases can use the following //EKYIN control statements:

- PROP LOAD
- PROP OFF
- TRACE
- TRDEST

PROP SUSP and RESIDENT control statements are ignored by MQ-ASYNC.

The TRACE and TRDEST //EKYIN control statements can be used in most other IMS DPROP utilities and components, including:

- The APPLY Program of MQ-ASYNC
- MVGU

- CCU
- SCU
- CUT
- GUU
- SCF Compare job
- SCF APPLY job
- PRDS Registration utility
- Job steps that invoke the RUP for asynchronous or user asynchronous propagation.

Figure 5 is an example of an //EKYIN DD statement that uses a trace control statement.

```
//EKYIN  DD   *
TRACE DEBUG=2 ;
/*
```

Figure 5. Example of //EKYIN DD Statement

//EKYLEOPT DD Statement

Use the //EKYLEOPT DD statement to provide Language Environment/370 (LE/370) run time options if:

- You installed LE/370.
- You wrote your IMS DPROP user exit routines in a high-level language that LE/370 supports.

The LE/370 run time options apply only to your IMS DPROP high-level language exit routines, and not to your propagating applications.

You can use the //EKYLEOPT statement in job steps where your exit routines execute, such as:

- RUP job steps
- CCU job steps
- DLU job steps

You can provide the LE/370 run time options on up to three records of the //EKYLEOPT DD statement. The statement must describe either a sequential OS/VS data set or a member of an OS/VS partitioned data set. The data set must have RECFM=F or FB, and LRECL=80.

Figure 6 and Figure 7 on page 7 show two examples using the //EKYLEOPT DD statement to specify run time options.

```
//EKYLEOPT DD *
RPTO(ON) RPTS(ON)
/*
```

Figure 6. Example 1 of //EKYLEOPT DD Statement

```
//EKYLEOPT DD *
RPTO(ON)
RPTS(ON) COUNTRY(US)
TRAP(OFF)
/*
```

Figure 7. Example 2 of //EKYLEOPT DD Statement

If LE/370 finds any errors in the run time options, it writes messages to the LE/370 message file (whose default DD name is SYSOUT). For more information on LE/370 run time options, see IBM SAA AD/Cycle Language Environment/370 Programming Guide.

//EKYPRINT DD Statement

Use the //EKYPRINT DD statement to print various messages.

The //EKYPRINT DD statement is a sequential OS/VS data set. Its characteristics, defined internally by IMS DPROP modules, are RECFM=FBA, LRECL=121, and BLKSIZE=1210. Figure 8 shows two examples of an //EKYPRINT DD statement.

```
//EKYPRINT   DD  SYSOUT=A
//EKYPRINT   DD  DSN=DPROP.EKYPRINT.JOBNAME,DISP=(MOD,CATLG),
//           UNIT=SYSDA,SPACE=(CYL,(1,1))
```

Figure 8. Example of //EKYPRINT DD Statement

When MPP/IFP regions invoke the RUP or the HUP for synchronous propagation, or EKYMQCAP for MQ-ASYNC propagation, the //EKYPRINT file can be opened and closed several times if either of the following occurs:
- Deadlocks are encountered
- The application abends and the region survives

Therefore, in MPP/IFP regions, you should define the //EKYPRINT file either as a SYSOUT data set, or with DISP=(MOD,...).

//EKYTRACE DD Statement

Use the //EKYTRACE DD statement to write IMS DPROP trace records for the following types of job steps:
- IMS DPROP utility job steps
- CCU, SCU, and Group Unload utility MVGU
- SCF Compare and APPLY utility **asynchronous**
- PRDS Registration utility
- Job steps used to call the RUP if you use a TRDEST control statement to set the trace destination to EKYTRACE
- IMS batch and dependent regions performing synchronous propagation or capturing changed data with MQ-ASYNC.
- APPLY Program of MQ-ASYNC **asynchronous**

The //EKYTRACE DD statement is a sequential OS/VS data set. Its characteristics, defined internally by IMS DPROP modules, are RECFM=FBA, LRECL=133, and BLKSIZE=3990.

Like the //EKYPRINT file, the //EKYTRACE file can be opened and closed several times during:

- Synchronous propagation in MPP/IFP regions, if the region encounters abends or deadlocks **synchronous**
- Capture with MQ-ASYNC in MPP/IFP regions, if the region encounters abends or deadlocks.
- Job steps that define propagation requests
- MVGU job steps

Therefore, define the //EKYTRACE file as a SYSOUT data set, or with DISP=(MOD,...).

Be sure to allocate sufficient space for the //EKYTRACE data set to accommodate all the IMS DPROP trace records. Figure 9 shows two examples of an //EKYTRACE DD statement. The first example shows the data set allocated as a SYSOUT data set. The second example shows a data set named DPROP.TRACE.JOBNAME allocated with DISP=MOD.

```
//EKYTRACE   DD  SYSOUT=A
//EKYTRACE   DD  DSN=DPROP.TRACE.JOBNAME,DISP=(MOD,CATLG),
//           UNIT=SYSDA,SPACE=(CYL,(50,50))
```

Figure 9. Examples of //EKYTRACE DD Statement

//EKYLOG DD Statement

You use the //EKYLOG DD statement if you used a TRDEST control statement to set the trace output destination to EKYLOG. The EKYLOG DD statement defines a data set to contain the trace records for the following types of job steps:

- IMS DPROP utility job steps, such as CCU, MVGU, and SCU
- Job steps used to call the RUP for asynchronous propagation **asynchronous**
- IMS batch and dependent regions performing synchronous propagation or capturing changed data with MQ-ASYNC
- APPLY Program of MQ-ASYNC

The data set named by //EKYLOG contains unformatted IMS DPROP trace records. Format these trace records with the IMS Log Formatting exit routine, EKYZ620X. For more information about this exit routine, see the *IMS DPROP Customization Guide*.

The //EKYLOG DD statement is a sequential OS/VS data set. Its characteristics, defined internally by IMS DPROP modules, are RECFM=VB, LRECL=4092, and BLKSIZE=4096.

The //EKYLOG file can be opened and closed several times during:

- Synchronous propagation in MPP/IFP regions, if the region encounters abends or deadlocks **synchronous**

- Capture with MQ-ASYNC in MPP/IFP regions, if the region encounters abends or deadlocks
- Job steps that define propagation requests
- MVGU job steps

Therefore, define the //EKYTRACE file as a SYSOUT data set, or with DISP=(MOD,...).

IMS DPROP can write numerous records to //EKYLOG. Be sure your JCL definitions allow for the creation of large //EKYLOG data sets.

Figure 10 shows an example of an //EKYLOG DD statement that uses DISP=MOD.

```
//EKYLOG     DD  DSN=DPROP.LOG.JOBNAME,DISP=(MOD,CATLG),
//           UNIT=SYSDA,SPACE=(CYL,(50,50))
```

Figure 10. Example of //EKYLOG DD Statement

//EKYSNAP DD Statement

Use the //EKYSNAP DD statement to write OS/VS SNAPs of the entire OS/VS task used to execute IMS DPROP. The SNAPs are created only if IMS DPROP encounters problems and if you have activated tracing for level 32 or level 64, usually upon request from IBM support personnel.

The //EKYSNAP DD statement is a sequential OS/VS data set. Its characteristics, defined internally by IMS DPROP modules, are RECFM=VBA, LRECL=125, and BLKSIZE=1632.

Figure 11 shows two examples of an //EKYSNAP DD statement. The first example shows the data set allocated as a SYSOUT data set. The second example shows a data set named DPROP.SNAP.JOBNAME allocated with DISP=MOD.

```
//EKYSNAP   DD  SYSOUT=A
//EKYSNAP   DD  DSN=DPROP.SNAP.JOBNAME,DISP=(MOD,CATLG),
//          UNIT=TAPE
```

Figure 11. Examples of //EKYSNAP DD Statement

The //EKYSNAP file can be opened and closed several times during:

- Synchronous propagation in MPP/IFP regions, if the region encounters abends or deadlocks **synchronous**
- Capture with MQ-ASYNC in MPP/IFP regions, if the region encounters abends or deadlocks
- Job steps that define propagation requests
- MVGU job steps

Therefore, define the //EKYTRACE file as a SYSOUT data set, or with DISP=(MOD,...).

IMS DPROP can write numerous records to //EKYSNAP. Be sure your JCL definitions allow for the creation of large //EKYSNAP data sets.

//EKYWTO DD Statement

Use the //EKYWTO DD statement to write console messages to a destination you specify on the //EKYWTO DD statement in addition to writing them to the operator console.

Chapter 2. //EKYIN Data Set control statements

The //EKYIN control statements are provided in the //EKYIN DD statement, which is described on page 5. This topic describes the control statements, their syntax, and five examples of their use. The control statements are:
- PROP LOAD (synchronous propagation and Capture of MQ-ASYNC)
- PROP OFF (synchronous propagation and Capture of MQ-ASYNC)
- PROP SUSP (synchronous propagation)
- RESIDENT (synchronous propagation)
- TRACE
- TRDEST

The PROP LOAD, PROP OFF, and PROP SUSP control statements are mutually exclusive.

The following topics provide additional information:

PROP LOAD

Use PROP LOAD control statements:
- when performing synchronous IMS-to-DB2 propagation
- when capturing IMS changes with MQ-ASYNC
- if you want to propagate IMS DB Insert calls performed in IMS jobsteps that load or reload an IMS database (if you want to propagate DB Insert calls performed with an IMS DB PCB with PROCOPT=L or PROCOPT=LS).

If you do not provide a PROP LOAD control statement, synchronous IMS-to-DB2 propagation and the Capture component of MQ-ASYNC ignore Insert calls performed in LOAD mode, and these inserts are therefore not propagated.

The PROP LOAD control statement does not affect DB2-to-IMS synchronous propagation because the DB2 Load utility never propagates to IMS.

In USER-ASYNC, the RUP always propagates DL/1 Insert calls, even those issued with PROCOPT=L or LS. If there are DL/1 insert calls with PROCOPT=L or LS that you do not want to propagate, do not call the RUP for the changes caused by these DL/1 insert calls.

By default, when the RUP is called to perform IMS-to-DB2 synchronous propagation, it does not propagate DL/I insert calls issued with PCBs that have a PROCOPT=L or PROCOPT=LS. If you include this PROP LOAD statement in //EKYIN, IMS DPROP ignores it.

Syntax of PROP LOAD

Figure 12 shows the syntax of the PROP LOAD control statement.

```
►►──PROP──LOAD──;──────────────────────────────────────────────────────►◄
```

Figure 12. The PROP LOAD control statement

Example of PROP LOAD

An example of the PROP LOAD control statement is shown in Figure 13.

```
PROP LOAD
```

Figure 13. Example of the PROP LOAD control statement

PROP OFF

If you have database repair job steps that must both update one copy of the data without triggering propagation, you can turn off synchronous propagation and the capture of MQ-ASYNC with the PROP OFF control statement of the //EKYIN data set.

To allow synchronous propagation to be turned off for selected jobsteps, call the SCU with the ALLOWPROPOFF control statement.

For the capture of MQ-ASYNC, you cannot use the SCU with an ALLOWPROPOFF control statement.

PROP OFF Syntax

Figure 14 shows the syntax of the PROP OFF control statement.

```
►►──PROP──OFF──;──────────────────────────────────────────────────────►◄
```

Figure 14. The PROP OFF control statement

Example of the PROP OFF control statement

An example of the PROP OFF control statement is shown in Figure 15.

```
PROP OFF
```

Figure 15. Example of the PROP OFF control statement

PROP SUSP

Use PROP SUSP to suspend synchronous propagation when you want to improve system performance, such as when running applications that perform a large number of updates to an HDAM database or a DEDB in physical RAP sequence. You should apply PROP SUSP to groups of data. For more information, see “Applying Propagation Request Status Changes to Groups of Data ” on page 185.

Before you can execute applications with the PROP SUSP control statement, you must set the propagation group status in the IMS DPROP directory to suspended using the SUSPEND control statement of the SCU. For information on SUSPEND, see page “SUSPEND ” on page 203 form=pageonly..

After you suspend propagation, however, you will eventually need to resynchronize the content of the IMS data copy and the DB2 data copy.

While processing an IMS update for a segment type, the RUP checks that the PROP SUSP control statement for the propagation groups propagating the segment type matches the propagation group status recorded in the IMS DPROP directory. Similarly, while processing a DB2 update for a table, the HUP checks that the PROP SUSP control statement for the propagation groups propagating the table match the propagation group status recorded in the IMS DPROP directory. If the specifications do not match, the RUP and HUP consider the resource to be unavailable, and back out the updates performed in the unit of work (UOW).

Table 2 identifies error conditions based on the //EKYIN control statement and the propagation group status in the IMS DPROP directory. This table describes the circumstances under which the RUP and HUP allow updates, and allow the updates to be propagated. The table assumes that the IMS DPROP system is not emergency stopped. If the IMS DPROP system is emergency stopped, all propagation stops and all propagation groups are ignored.

Table 2. Error Conditions Based on //EKYIN control statements and Propagation Group Status

//EKYIN Control Statements	Propagation Group Status in IMS DPROP Directory	Updates Allowed	Updates Propagated
PROP SUSP	Active	No/backout	-
	Suspended	Yes	No
	Inactive	Yes	No
PROP OFF	Active	Yes[1]	No
	Suspended	Yes[1]	No
	Inactive	Yes	No
No statement provided	Active	Yes	Yes
	Suspended	No/backout	-
	Inactive	Yes	No

[1]Updates are allowed if the SCU has been executed with an ALLOWPROPOFF control statement.

The checking performed by the RUP and HUP protects against:

- Concurrent execution of updates by jobs with PROP SUSP control statements and updates by jobs performing normal synchronous propagation without PROP SUSP control statements. This type of concurrent execution would result in synchronous propagation failures and inconsistent data.
- Inadvertent execution without propagation of job steps requiring synchronous propagation.

IMS DPROP provides protection in two ways:

- If you use PROP SUSP control statements for a job step, during the first change to an affected segment or table, the RUP or HUP ensure that all propagation groups that are propagating this segment or table have either PROP SUSP or INACTIVE status in the IMS DPROP directory. If the propagation group status is other than PROP SUSP or INACTIVE, the RUP or HUP consider this a "resource-unavailable" situation and back out the updates performed by the UOW.
- If you do *not* use PROP SUSP or PROP OFF control statements, then during the first change to a segment or table, the RUP or HUP ensure that no propagation group that is propagating the segment or table has PROP SUSP status in the IMS DPROP directory.

For more information on this type of protection and on "resource-unavailable" logic, refer to the appropriate Administrators Guide for your propagation mode.

Syntax of the PROP SUSP control statement

Figure 16 shows the syntax of the PROP SUSP control statement.

```
►►──PROP──SUSP──────────────────────────────────────────────────────────►
►──┬─ALL──────────────────────────────────────────────────────────────┬──;──────►◄
   │                ┌─,─┐                                             │
   ├─DBD=(──────dbd─┴───)──────────────────────────────────────────────┤
   │                ┌─,─────────────┐                ┌─,─────────────┐│
   ├─DBD=dbd,SEG=(──┴──segment──── delim end.)─propagation group=(──┴──propagation group──id──┴──)─┤
   │                ┌─,─┐                                             │
   └─,PRSET=(───────┴─prset──┴──)──┘
```

Figure 16. Syntax of the PROP SUSP control statement

ALL
Indicates that the synchronous propagation of all updates performed by the job step should be suspended.

DBD=(*database1,database2,...* **)**
Lists the physical databases for which synchronous propagation of updates from and to the physical databases should be suspended.

For IMS-to-DB2 synchronous propagation, IMS DPROP suspends propagation *from* the listed IMS databases.

For DB2-to-IMS synchronous propagation, IMS DPROP suspends propagation *to* the listed IMS databases.

DBD= *database* **,SEG=(***segname1,segname2,...***)**
Indicates that the synchronous propagation of the updates from and to the segments you have listed should be suspended.

Do not use the SEG= keyword to identify internal segments[1].

To suspend propagation groups that synchronously propagate internal segments, you can specify either:
- The propagation group IDs on the propagation group= keyword
- The name of the containing IMS segment on the SEG= keyword

If you specify the name of the containing IMS segment on the SEG= keyword, then the SCU SUSPEND control statement applies to all propagation groups propagating from or to:
- The internal segments
- The containing IMS segment

If you used the SEG= keyword to identify an IMS segment that contains fields used as PATH data, then:
- The SUSPEND control statement is applied *only* to those propagation groups that are synchronously propagating:
  - the identified segment as an entity or extension segment
  - internal segments contained in the identified IMS segment
- The SUSPEND control statement is *not* applied to those propagation groups that are both synchronously propagating dependent segments and including path data from the identified segment.

propagation group=(*prid1,prid2,...* **)**
Lists propagation groups for which synchronous propagation should be suspended.

PRSET=(*setid1,setid2,...* **)**
Lists the propagation group sets for which to suspend propagation.

Examples of the PROP SUSP control statement

Figure 17 shows examples of the PROP SUSP control statement.

```
PROP SUSP ALL
PROP SUSP DBD=(SKILLDB,INVENTRY);
PROP SUSP ALL, PRSET=PRSET1;
```

Figure 17. Examples of the PROP SUSP control statement

RESIDENT

In some cases, you can use the RESIDENT control statement in MPP regions to influence the performance of synchronous propagation.

For IMS-to-DB2 synchronous propagation, you can use the SQLU= and PRCB= keywords to reduce the performance impact associated with loading the SQL update modules and with reading the RUP propagation control blocks (PRCBs) from virtual lookaside facility (VLF).

1. Internal segments are not IMS segments; instead they represent structures embedded within a containing IMS segment and propagated by mapping case 3 propagation groups.

For DB2-to-IMS synchronous propagation, you can use the HPRCB= keywords to reduce the performance impact associated with reading the HUP PRCBs from VLF.

Use the RESIDENT control statement to control:

- The number '*n1*' of most recently used HUP PRCBs to remain resident in the virtual storage of each MPP region (for DB2-to-IMS synchronous propagation only).
- The number '*n2*' of most recently used RUP PRCBs to remain resident in the virtual storage of each MPP region (for IMS-to-DB2 synchronous propagation only).
- The number '*n1*' of most recently used SQL update modules to remain resident in the virtual storage of each MPP region (for IMS-to-DB2 synchronous propagation only). IMS DPROP maintains the SQL update modules and PRCBs on a "most recently used" chain (MRU). In terms of performance, it is more important to maintain enough SQL update modules than PRCBs in virtual storage.

  You can change the default value with the RESIDENT control statement. If you increase the value you can:

  – Reduce the elapsed time and CPU time required to load the SQL update modules and read the PRCBs in MPP regions.

  – Increase the virtual storage requirements and paging in MPP regions.

  If you decrease the value you can:

  – Increase the elapsed time and CPU time required to load the SQL update modules and read the PRCBs in MPP regions.

  – Reduce the virtual storage requirements and paging in MPP regions.

If you have a test system with a low transaction rate of synchronous MPPs, you can request that zero SQL update modules and PRCBs be maintained in MPP regions by using SQLU and RPRCB control statement parameters.

Syntax of the RESIDENT control statement

Figure 18 shows the syntax of the RESIDENT control statement.

```
                  ┌─,──────────────┐
                  ▼                │
►►──RESIDENT──────┬─HPRCB=nn───────┴──;──────────────────────────────────►◄
                  ├─RPRCB=nn───┤
                  └─SQLU=nn────┘
```

Figure 18. The RESIDENT control statement

HPRCB= *nn*
Specifies how many HUP PRCBs should be maintained on the MRU chain. *nn* can be any value between 0 and 999999999, inclusive. The value of *nn* does not include the HUP PRCBs used for the currently executing MPP. Those PRCBs are always kept in virtual storage, even if you specify HPRCB=0. *nn* specifies the number of HUP PRCBs used during previous MPP executions that should be maintained in virtual storage. The default value is 20.

RPRCB= *nn*
Specifies how many RUP PRCBs should be maintained on the MRU chain. *nn* can be any value between 0 and 999999999, inclusive. The value of *nn* does

not include the RUP PRCBs used for the currently executing MPPs. Those PRCBs are always kept in virtual storage, even if you specify PRCB=0. *nn* specifies the number of RUP PRCBs used during previous MPP executions that should be maintained in virtual storage. The default value is 20.

SQLU= *nn*
Specifies how many SQL update modules should be maintained on the MRU chain. *nn* can be any value between 0 and 999999999, inclusive. The value of *nn* does not include the SQL update modules:
- used for the currently executing MPPs
- associated with resident RUP PRCBs

Those modules are always kept in virtual storage, even if you specify SQLU=0. *nn* specifies the number of SQL update modules used during previous MPP executions that are not associated with resident RUP PRCBs and that should be maintained in virtual storage. The default value is 40.

Examples of the RESIDENT control statement

Examples of the RESIDENT control statement are shown in Figure 19.

```
RESIDENT SQLU=75,RPRCB=50
RESIDENT HPRCB=50;
RESIDENT SQLU=75,RPRCB=50,HPRCB=40;
```

Figure 19. Examples of the RESIDENT control statement

TRACE

Use the TRACE control statement to trace IMS DPROP utility functions and synchronous and asynchronous propagation performed by the:
- Capture Component of MQ-ASYNC
- APPLY Program of MQ-ASYNC
- RUP
- HUP (synchronous)
- Receiver (asynchronous)
- Selector (asynchronous)

You can trace the propagating activities of the RUP and the HUP with either:
- TRACE control statements in the //EKYIN data set of the propagating batch or dependent region
- The SCU TRACEON control statement

The TRACE control statement traces the activities of *one* specific IMS batch or dependent region. The SCU TRACEON control statement traces the activities of all propagating MS batch and dependent regions.

Syntax of the Trace control statement

Figure 20 on page 18 shows the syntax of the TRACE control statement to trace the synchronous and asynchronous activities of the:
- Capture of MQ-ASYNC
- APPLY Program of MQ-ASYNC
- Receiver
- RUP
- HUP

```
►►─TRACE──DEBUG=n──┬──────────────────────────────────────────────────────────►
                   │                  ┌─,──────────┐
                   ├─,DBD=(──────────▼──dbdname──────┴──)──────────────────────┤
                   │                  ┌─,──────────┐
                   └─,DBD=dbdname»»,SEG=(──────────▼──segname──────┴──)─────────┘

►──┬──────────────────────────────────┬──;──────────────────────────────────►◄
   │                  ┌─,──────────┐  │
   └─,PSB=(──────────▼──psbname──────┴──)──┘
```

Figure 20. The TRACE control statement for the Activities of the RUP

Figure 21 shows the format of the TRACE control statement to trace IMS DPROP utility functions such as:

- MVGU
- SCU
- CCU
- DLU synchronous
- GUU asynchronous
- SCF Compare asynchronous
- SCF Apply asynchronous

```
►►─TRACE──DEBUG=n──;──────────────────────────────────────────────────────────►◄
```

Figure 21. The TRACE control statement for IMS DPROP Utility Functions

DEBUG= *n*

Provides different traces for:

- synchronous IMS-to-DB2 propagation with the RUP
- Capture of MQ-ASYNC
- asynchronous IMS-to-DB2 propagation with the Receiver or with the Apply program
- synchronous DB2-to-IMS propagation with the HUP
- CCU executions

Levels 0, 1, 2, 4, 8, 16, 32, and 64 are defined. Additionally, you can request a combination of tracing levels by adding the values of the desired trace levels. For example, DEBUG=7 provides tracing levels 1, 2, and 4. In “Examples of the TRACE control statement ” on page 21, DEBUG=30 provides tracing levels 2, 4, 8, and 16. Therefore, *n* can be any integer between 0 and 127. For more information on specifying DEBUG levels, refer to IMS DPROP Diagnosis.

The trace levels are:

Level	Description
0	Provides the same trace as level 1.
1	Provides an internal wraparound trace showing module entries and exits, including user exit routines.

This level of trace is always performed whether you request it or not.

2 Provides different traces for IMS-to-DB2 propagation with the RUP, IMS-to-DB2 propagation with the Receiver, and CCU executions.

- For IMS-to-DB2 propagation with the RUP, level 2 provides an external trace of the changed IMS data and of the propagating SQL update statements. The trace of the changed IMS data includes:
  - The changed data IMS segments with their fully concatenated keys
  - The IMS segment name
  - The database name
  - The type of update

  The trace of the SQL update modules includes:
  - The table name
  - The description of any SQL WHERE clause including column names and column contents
  - The description of all propagated columns including column names and column contents
  - The error code
  - The type of update

  For the Capture of MQ-ASYNC and for the Apply program of MQ-ASYNC, level 2 also include a trace of the MQSeries operations and of the MQSeries messages.

  For IMS-to-DB2 propagation with the Receiver, level 2 also provides:
  - A list of all propagation requests that are active for the current execution of the Receiver.
  - The value of each unit of work recovery token at which a DB2 commit is issued.
- DB2-to-IMS synchronous propagation with the HUP, and of the propagating IMS calls. The trace of the changed DB2 data includes:
  - Changed DB2 rows
  - SQLDAs describing the columns of the changed rows
  - DB2 table name
  - Type of update

  The trace of the synchronously propagating IMS calls includes:
  - Type of IMS call
  - PCB status code
  - IMS segment name
  - DBD name
  - IMS segments with their fully concatenated key
  - SSAs used in the IMS calls
- For CCU executions, level 2 provides an external trace of the compared IMS segments and DB2 rows. The segment trace includes the IMS segment data, the fully concatenated key, the segment name, and the DBD name. The DB2 row trace includes the SQLDA describing the DB2 rows and column values.

4 Provides an external trace of the calls with all interface information, including changed data and interface control blocks, to the following :

- RUP
- HUP
- EKYMQCAP Capture component of MQ-ASYNC

For more information on interface control blocks, see*Diagnosis*.

A level 4 trace also provides information on calls to Segment exit routines, Field exit routines, and Propagation exit routines, including the input and output of the exits.

8 Provides an external trace of entries and returns for modules and major routines.

16 Provides an external trace of activities and control blocks of the:
- HUP (synchronous)
- RUP
- IMS DPROP service functions (most IMS DPROP tracing is at level 16)
- Selector and Receiver
- EKYMQCAP Capture component and the Apply program of MQ-ASYNC
- Other IMS DPROP components, except the DLU, CCU, MVG, MVGU, and MCE

32 Provides an external trace of the snapping of the entire OS/VS subtask to the //EKYSNAP file when IMS DPROP detects problems that do not prevent IMS DPROP from continuing its processing.

The LOG-ASYNC propagation Selector writes some additional detailed memory allocation and deallocation tracing at level 32.

For each Selector or Receiver execution, level 32 provides detailed tracing associated with the processing of IMS log record type X'9904', changed data capture record.

64 Provides an external trace of activities and control blocks of the DLU, MVGU, CCU, MVG, and MCE.

DBD=(*dbdname,dbdname,...* **)**
Limits the trace output to the propagation activities of the databases you list.

For IMS-to-DB2 propagation, IMS DPROP traces propagation *from* the listed IMS databases.

For DB2-to-IMS synchronous propagation, IMS DPROP traces propagation *to* the listed IMS databases.

DBD= *database* **,SEG=(***segment1,segment2,...***)**
Limits the trace output to the propagation activities from and to the physical IMS segments you list.

Do not use the SEG= keyword to identify internal segments. Internal segments are not IMS segments; they represent structures embedded within a containing IMS segment and are propagated by mapping case 3 propagation groups.

Instead, when requesting tracing for propagation groups propagating internal segments, specify the name of the containing IMS segment in the SEG= keyword. If you specify the name of the containing IMS segment, the TRACE control statement applies to all propagation requests propagating from or to the internal segments and containing IMS segment.

PSB=(*psbname,psbname,...* **)**
Limits the trace output to the propagation activities of the PSBs you list.

If you do not want to limit the output to a specific segment, database, or PSB, then the DEBUG level you specify applies to all segments, databases, and PSBs.

You can provide multiple TRACE control statements. You can also activate a trace with both the TRACE control statement and with the SCU control statement TRACEON. Therefore, it is possible to have multiple DEBUG specifications for a PSB, a database, or a segment type. In such a case, the combination of the DEBUG levels prevails.

Examples of the TRACE control statement

Figure 22 shows examples of the TRACE control statement.

```
TRACE DEBUG=2
TRACE DEBUG=30;
```

Figure 22. Examples of the TRACE control statement

TRDEST

Use the TRDEST control statement to override the default destination of the IMS DPROP trace output. The TRDEST control statement enables you to direct the trace to:

- The IMS log
- The //EKYLOG DD statement
- The //EKYTRACE DD statement

Trace output written to:

- The //EKYTRACE DD statement is already formatted.
- The IMS log or to the //EKYLOG data set is unformatted. Therefore, it results in fewer I/O operations, less external storage, and a smaller CPU overhead for the traced job step.

The IMS DPROP defaults for the trace destinations are:

- The IMS Log (for synchronous propagation and for the Capture component of MQ-ASYNC)
- The //EKYLOG data set (for LOG-ASYNC and for the Apply program of MQ-ASYNC)
- The //EKYTRACE data set for IMS DPROP utility functions

Syntax of the TRDEST control statement

Figure 23 shows the syntax of the TRDEST control statement.

```
                    ┌─,──────────────┐
                    ▼                │
►►─TRDEST─DEST=(────┬─IMSLOG───┬──────┴─)─;──────────────────────────────►◄
                    ├─EKYLOG───┤
                    └─EKYTRACE─┘
```

Figure 23. The TRDEST control statement

DEST=
Specifies where the external trace records are written.

IMSLOG
Writes external trace records to the IMS log. IMSLOG is only valid for

TRDEST control statements when performing synchronous propagation, or when capturing changes with MQ-ASYNC.

You format external trace records written to the IMS log with the IMS Log Formatting exit routine, EKYZ620X.

EKYLOG
Writes the external trace records to the //EKYLOG DD statement.

The JCL of the traced job step should provide an //EKYLOG DD statement. Also, the JCL definition of the traced job step should provide for large //EKYLOG files for all of the created trace records. Trace levels greater than 2 result in a substantial increase in the number of trace records written to the //EKYLOG data set.

Format external trace records written to the //EKYLOG data set with the IMS Log Formatting exit routine, EKYZ620X. For more information on this exit routine, see*IMS DPROP Diagnosis*.

EKYTRACE
Writes the external trace records to the //EKYTRACE DD statement. The JCL of the IMS batch or dependent region should provide an //EKYTRACE DD statement. Also, the JCL definitions of the IMS batch or dependent regions should provide for large //EKYTRACE files for all of the created trace records. Trace levels greater than 2 result in a substantial increase of the number of trace records written to the //EKYTRACE data set.

Example of the TRDEST control statement

An example of the TRDEST control statement is shown in Figure 24.

```
TRDEST DEST=(IMSLOG,EKYTRACE)
```

Figure 24. Example of the TRDEST control statement

How to Format and Print //EKYLOG Trace Records

You can selectively format and print the trace records you write to the //EKYLOG data set, or to the IMS log. To do so, use EKYZ620Z, a IMS DPROP formatting and printing routine that executes as an exit to the IMS file select and formatting print utility, DFSERA10. Use the DFSERA10 JCL and control statements to select IMS DPROP trace records from the IMS log or from //EKYLOG. On these control statements, you code an EXITR keyword that specifies EKYZ620X.

For additional information on:
- EKYZ620X, see *IMS DPROP Diagnosis*
- DFSERA10, see *IMS/ESA® Utilities Reference: System*

You must format the IMS log records or //EKYLOG records with EKYZ620X and DFSERA10 on an MVS system where IMS is installed.

Part 2. Defining Propagation Requests

Chapter 3. Defining Propagation Requests Using DataRefresher

The DataRefresher UIM (user input manager) commands and control statements allow you to define a IMS DPROP propagation request (PR). You can use DataRefresher UIM to define a propagation request even if you do not intend to perform an extract with DataRefresher. For example, you can use DataRefresher UIM to define a propagation request performing one-way DB2-to-IMS synchronous propagation.

The following topics provide additional information:

- "DataRefresher UIM Commands that Define Propagation Requests " on page 26
- "JCL for Defining Propagation Requests with DataRefresher UIM " on page 75
- "DataRefresher DEM Commands " on page 84
- "JCL to Extract Data with DataRefresher DEM " on page 86

Refer to the appropriate DataRefresher or DXT documentation for a complete description of DataRefresher commands. Unless otherwise noted, the commands described in this topic apply to the generalized mapping cases and to user mapping cases. For more information about mapping cases, refer to the appropriate *Administrators Guide* for your propagation mode.

DataRefresher UIM Commands that Define Propagation Requests

This topic describes the DataRefresher UIM commands and statements that are required to define a propagation request. These include:

- CREATE DATATYPE command
- CREATE DXTPSB command
- CREATE DXTVIEW command
- SUBMIT command with its associated EXTRACT statement

For more information on naming conventions, see *MVS/ESA JCL Reference*.

CREATE DATATYPE

Use CREATE DATATYPE to create a new user-defined data type description.

CREATE DATATYPE commands are required only if you:

- Intend to use Field exit routines because your IMS database contains fields in a format not supported by IMS DPROP or DataRefresher
- Want to perform data conversions not directly supported by IMS DPROP or DataRefresher

Each CREATE DATATYPE command defines a unique two-character name for a user data type and associates a Field exit routine to this user data type.

Figure 25 shows the syntax of the CREATE DATATYPE command.

```
►►──CREATE──DATATYPE──SRCTYPE=usertype──,EXIT=exitname──,SRCBYTES=──┬─n──────┬──►
                                                                    └─VARIES─┘

►──┬──────────────────────┬──,TRGTYPE=dxt_data_type──,TRGBYTES=──┬─n──────┬──►
   └─,SRCSCALE=──┬─n──────┬─┘                                    └─VARIES─┘
                 └─VARIES─┘

►──┬──────────────────────┬──┬──────────────────────┬──;──────────────────────►◄
   └─,TRGSCALE=──┬─n──────┬─┘  └─,DESC='comment'─┘
                 └─VARIES─┘
```

Figure 25. The CREATE DATATYPE Command

CREATE DATATYPE (REQUIRED)
Specifies that a user data type is being created.

SRCTYPE= *usertype* **(REQUIRED)**
Specifies a two-alphanumeric character value used to uniquely identify this user data type.

Replace *usertype* with a unique value among both standard DataRefresher data types and other user data types. *usertype* must be two nonblank characters and must not be *VC* or *VG*.

EXIT= *exitname* **(REQUIRED)**
Identifies the name of the exit routine load module used to convert the source format associated with the data type into the target format supported by DataRefresher and by IMS DPROP.

Replace *exitname* with the name of the Field exit routine you want to invoke for this user data type.

SRCBYTES= *n* **| VARIES (REQUIRED)**
Specifies the length (in bytes) of the source field if *n* is specified; or indicates that the number of bytes can vary (VARIES).

The values you can specify are:

n Replace *n* with the length, in bytes, of the fixed length user data type. This value must be a number from 1 through 32767.

VARIES Indicates that the fields of this user data type can have different BYTES values coded in the FIELD statement of the CREATE DXTPSB command.

SRCSCALE= *n* **| VARIES (OPTIONAL)**
Specifies the scale of the source field if *n* is specified; or indicates that the scale can vary (VARIES).

The values you can specify are:

n Replace *n* with the scale of the user data type. This value can be an integer from 0 through 255.

VARIES Indicates that the fields of this user data type can have different SCALE values coded in the FIELD statement of the CREATE DXTPSB command.

TRGTYPE= *dxt_data_type* **(REQUIRED)**
Specifies to which DataRefresher data type the user data type will be converted.

Replace *dxt_data_type* with a one- or two-character DataRefresher data type:

A Date data
B Single-byte binary data
C EBCDIC character string
D Long (8-byte) floating-point number
E Short (4-byte) floating-point number
F 32-bit, signed binary integer
G Only DBCS data
H 16-bit, signed binary integer
P Packed decimal number
S Time stamp data
T Time data
VC Variable-length character data
VG Variable-length graphic data
Z Zoned decimal number

The date and time format returned by your Field exit routine must be in one of the following formats: ISO, JIS (Japanese Industrial Standard), USA (IBM USA Standard), or EUR (IBM European Standard).

TRGBYTES= *n* **| VARIES (OPTIONAL)**
Specifies the length (in bytes) of the target field (in converted format) if *n* is specified; or indicates that the number of bytes can vary (VARIES).

The values you can specify are:

n Indicates the length of the target field. Replace *n* with the length of the data that this exit will return.

VARIES Specifies that the maximum target field length will be established for each field using this data type by the definition (DEF) call.

TRGSCALE= *n* **| VARIES (OPTIONAL)**
Specifies the scale of the target field (in converted format) if *n* is specified; or indicates that the scale can vary (VARIES). TRGSCALE is only applicable when TRGTYPE is P or Z.

Fields of type P and Z represent numbers in packed or zoned decimal format, a representation that can be interpreted in many different ways because it does not contain an explicit decimal point.

The values you can specify are:

n If you want the decimal point to be placed to the left of the last *n* digits in the decimal representation. Replace *n* with a positive integer that specifies the scale of the data this exit will return.

The value of *n* can range from 0 through the number of decimal digits that the described field contains. For an *n*-byte field of type Z, the maximum scale is *n*. For an *n*-byte field of type P, the maximum scale is *2n-1*. The default for *n* is 0.

VARIES Indicates that the target scale will be established for each field using this data type by the DEF call.

DESC=’ *comment* **’ (OPTIONAL)**
Saves a comment in the DataRefresher FDTLIB about the user data type you are creating. For more information about coding this keyword, see Appendix A, “Coding the Description ,” on page 591.

CREATE DXTPSB

Use CREATE DXTPSB to describe to DataRefresher your IMS databases, segments, and fields. You normally describe an IMS database to DataRefresher only once.

If you intend to use DataRefresher to define IMS DPROP propagation requests, you must distinguish between a propagation request that:
- Belongs to a generalized mapping case
- Belongs to a user mapping case

The mapping supported by the generalized mapping logic of IMS DPROP is a subset of the more complex mapping supported by DataRefresher. Therefore, the CREATE DXTPSB definitions required to create IMS DPROP propagation requests belonging to a generalized mapping case are a subset of the full DataRefresher definition capabilities.

The description of the CREATE DXTPSB command in the next topic focuses on that subset of keyword and keyword values used to define a propagation request belonging to a generalized mapping case. This topic does not describe DataRefresher facilities that are *not supported* by the generalized mapping logic of IMS DPROP (for example, fields with a variable start position, and date/time conversion routines).

For information on using this command for user mapping cases, refer to “Using CREATE DXTPSB for User Mapping Cases” on page 44. For a complete description of the full set of keywords and keyword values that can be used to

define a propagation request belonging to a user mapping case, refer to the appropriate DataRefresher or DXT documentation for a complete description of DataRefresher commands.

Using CREATE DXTPSB for Generalized Mapping Cases

The CREATE DXTPSB command consists of:

- The description of the DXTPSB itself.
- One or more DXTPCB statements, each describing an IMS database. Typically a DXTPSB will have only one DXTPCB.
- One or more SEGMENT statements describing the segments of each DXTPCB. You need to distinguish between:

  SEGMENT statements describing IMS segments
  Provide one SEGMENT statement for each propagated IMS database segment and for the segment's physical parent and ancestors (up to the root segment).

  SEGMENT statements describing internal segments
  Provide one SEGMENT statement for each internal segment propagated by mapping case 3 propagation requests.

  Internal segments are not IMS segments. Internal segments are embedded structures located within an IMS segment. A typical example of an embedded structure is a *repeating group* of fields with a fixed or variable number of occurrences within the IMS segment. See "SEGMENT Statement for Internal Segments " on page 34 for more information on internal segments.

  Provide the SEGMENT statements in the hierarchical order of the segments (from top to bottom and left to right).
- One or more FIELD statements describing the fields of each segment.

The syntax of CREATE DXTPSB used with a propagation request belonging to a generalized mapping case is shown in Figure 26 on page 30.

```
►►──CREATE──DXTPSB──NAME=psbname──┬──────────────────────────────────────►
                                  └─,DESC='comment'─┘

►──┤ DXTPCB Statement ├──┬─┤ SEGMENT Statement (Version 1) ├──┬─┤ FIELD Statement ├──;──►◄
                         ├─┤ SEGMENT Statement (Version 2) ├──┤
                         └─┤ SEGMENT Statement (Version 3) ├──┘
```

DXTPCB Statement:

```
├──DXTPCB──NAME=pcbname──┬──────────────────┬──DBACCESS=dbtype──┬──────────────────┬──┤
                         └─,DBNAME=dlidb────┘                   └─,DESC='comment'──┘
```

SEGMENT Statement Version 1 (For IMS segments not processed by a Segment exit routine):

```
├──SEGMENT──NAME=segname──,PARENT=──┬─0──────────┬──,BYTES=n──┬──────────────────────────┬──┤
                                    └─psegname───┘            │ ┌─F─┐                    │
                                                              ├─,FORMAT=──┤   ├──────────┤
                                                              ├─,MAXNBR=──1──────────────┤
                                                              ├─,FREQ=──n────────────────┤
                                                              └─,DESC=──'comment'────────┘
```

SEGMENT Statement Version 2 (For IMS segments processed by a gment exit routine):

```
├──SEGMENT──NAME=segname──,PARENT=──┬─0──────────┬──,BYTES=n──┬──────────────────────────┬──►
                                    └─psegname───┘            │ ┌─F─┐                    │
                                                              ├─,FORMAT=──┤   ├──────────┤
                                                              ├─,MAXNBR=──1──────────────┤
                                                              ├─,FREQ=──n────────────────┤
                                                              └─,DESC=──'comment'────────┘

►──┬─,DATAEXIT=──exitname──┬──,XBYTES=n──┬──────────────────────────┬──┤
   └─,EXIT=──exitname──────┘             │ ┌─N─┐                    │
                                         └─EODCALL=──┤   ├──────────┘
```

SEGMENT Statement Version 3 (For internal segments):

```
├──SEGMENT──NAME=segname──,PARENT=psegname──,FORMAT=──┬─FI─┬──,OCCURS=──┬─n──────────┬──►
                                                      └─VI─┘            └─fieldname──┘

►─,START=n──┬──────────────┬──┬──────────────────┬──┬──────────────┬──┬──────────────┬──┤
            ├─fieldname+n──┤  ├─,BYTES=──n───────┤  ├─,FREQ='n'───┤  ├─,DESC='comment'─┤
            └─segname+n────┘  └─,NEXT=──fieldname+n┘  └──────────────┘  └──────────────┘
```

FIELD Statement:


```
├──FIELD──NAME=fieldname──,START=n──┬──────────────────────────────────────────┬──┤
                                    │ ┌─F─┐                                    │
                                    ├─,FORMAT=──┤   ├──────────────────────────┤
                                    ├─,MAXNBR=──1──────────────────────────────┤
                                    ├─,FREQ=──n────────────────────────────────┤
                                    ├─,DESC=──'comment'────────────────────────┤
                                    ├─,DATAEXIT=──exitname─────────────────────┤
                                    ├─,EXIT=──exitname─────────────────────────┤
                                    ├─,XBYTES=n────────────────────────────────┤
                                    │ ┌─N─┐                                    │
                                    └─EODCALL=──┤   ├──────────────────────────┘
```

CREATE DXTPSB (REQUIRED)
Specifies that a DXTPSB description is being created.

NAME= *psbname* **(REQUIRED)**
Assigns a name to the DXTPSB description. Replace *psbname* with the name. The name can contain alphanumeric or special characters and must be unique among all DXTPSB names in the FDTLIB.

The name of your DXTPSB can match the name of its corresponding IMS PSB, but it does not need to match. Therefore, you can include any number of different descriptions of the same PSB in an FDTLIB.

Be sure to give the PSB name to the person who writes the JCL for running the DEM. That way, the DEM can process extract requests that reference the DXTPSB. If your site uses RACF to protect its DataRefresher data items, you must also give the PSB name to the RACF administrator.

DESC=’ *comment* **’ (OPTIONAL)**
Saves a comment in the DataRefresher FDTLIB about the data description you are creating. For more information about coding this keyword, see Appendix A, “Coding the Description ,” on page 591.

DXTPCB Statement

DXTPCB (REQUIRED)
Describes database PCBs. You can write more than one PCB for the PSB you are describing. Write them in the order in which their PCBs are defined in the IMS PSB.

IMS DPROP restrictions associated with PCBs in the IMS PSB are:

- The PCBs should reference physical IMS DBDs.
- The PCB statement in the IMS PSB should not request access (with the PROCSEQ= keyword) through a secondary index, which would result in a restructure of the hierarchical structure.
- Field sensitivity should not be specified.

If these rules are not observed, the mapping performed by DataRefresher during the extract and the mapping performed by IMS DPROP will probably be inconsistent, resulting in propagation failures.

NAME= *pcbname* **(REQUIRED)**
Assigns a name to the PCB being described. Replace *pcbname* with the name. The name can contain alphanumeric or special characters, and must be unique among the names assigned to the PCBs included in this DXTPSB description.

DBACCESS= *dbtype* **(REQUIRED)**
Describes the type of database. The values supported by IMS DPROP are:

HISAM	Hierarchical Indexed Sequential Access Method
SHISAM	Simple Hierarchical Indexed Sequential Access Method
HIDAM	Hierarchical Indexed Direct Access Method
HDAM	Hierarchical Direct Access Method
DEDB	Data Entry databases (Fast Path)

IMS DPROP, DXT Version 2 Release 5, and DataRefresher also support a value of DBACCESS=HSSR for HISAM, SHISAM, HIDAM, and HDAM databases. With this value, DXT and DataRefresher use High Speed Sequential Retrieval (HSSR) to read the IMS database during the extract. Using HSSR for

the extract typically results in faster extracts. DXT and DataRefresher have various restrictions for DBACCESS=HSSR. Refer to the appropriate DataRefresher or DXT documentation for a description of these restrictions.

IMS DPROP does not support MSDBs, HSAM, SHSAM, or GSAM databases.

DESC=’ *comment* **’ (OPTIONAL)**
Saves a comment in the DataRefresher FDTLIB about the data description you are creating. For more information about coding this keyword, see Appendix A, “Coding the Description ,” on page 591.

DBNAME= *dlidb* **(REQUIRED unless the name of the DXTPCB is the name of the physical IMS DBD)**
Specifies the IMS database name as defined in your DBD. It must be a physical database. IMS DPROP requires that the DBD exist in the DBDLIB.

DBNAME can be an alphanumeric name from 1 to 8 characters. The default for DBNAME is the DXTPCB name. If the DXTPCB name is the name of the physical DBD, IMS DPROP accepts the default.

SEGMENT Statement for IMS Segments

The SEGMENT statements define both IMS segments and internal segments (repeating groups within the IMS segments). They describe to DataRefresher the hierarchical structure of the IMS segments and the internal segments.

Write the SEGMENT statements in hierarchical order (from top to bottom and left to right). For generalized mapping, you must include a SEGMENT statement for each propagated segment and for the segment’s physical parents and ancestors (up to the root segment).

For a description of SEGMENT statements used to define internal segments, refer to “SEGMENT Statement for Internal Segments ” on page 34.

SEGMENT (REQUIRED)
Use one SEGMENT for each database segment involved in propagation. For a DEDB PCB, you can write up to 127 database segments; for all other PCBs, you can write up to 255 segments.

You can include in the DXTPCB a SEGMENT statement describing the sequential dependent (SDEP) segment of a DEDB. But, the SDEP can only be propagated through user mapping (PRTYPE=U), not by the generalized mapping logic of IMS DPROP.

Database segments can be fixed or variable in length and can contain variable-length fields. These variable-length fields can be either variable-length character (VC) or variable-length graphic (VG) data types.

NAME= *segname* **(REQUIRED)**
Names the IMS database segment.

Replace *segname* with the IMS name of the segment type, that is, the name that was used for the segment in the physical DBD generation.

PARENT= *0* | *psegname* **(REQUIRED)**
Indicates the parent of the segment. The values you can specify are:

0 If the segment being described is a root segment

psegname
If the segment is not a root segment. Replace *psegname* with the name of the parent segment.

BYTES= *n* **(REQUIRED for database segments)**

Specifies the length of the segment in bytes.

For a fixed-length segment, replace *n* with the length of the segment, in bytes.

For a variable-length segment, replace *n* with the maximum length of the segment, in bytes. Include the data portion and the 2-byte length field for the segment.

For segments associated with a Segment exit routine, use BYTES to define the length of the segment to IMS. During IMS-to-DB2 propagation, this is the length of the segment as retrieved from IMS, *before* processing by the exit.

The value of *n* comes from the BYTES keyword of the related DBDGEN SEGMENT command. Be sure to include 2 for the length field of a variable-length segment (as the BYTES keyword does for DBDGEN).

The value of *n* reflects the length of the segment as it appears in the I/O area of a DL/I call using a DL/I PCB, which references a physical DBD and which does not use field sensitivity.

For a logical child segment, this length includes the length of the fully concatenated key of the logical parent (as does the BYTES keyword of the DBDGEN). This is so even if the key of the logical parent is defined as “virtual” in the DBDGEN.

FORMAT= F | V (OPTIONAL for database segments)

Indicates the format of the segment. If the segment is processed by a Segment exit routine, this keyword describes the format of the segment as it is defined to IMS DPROP. During IMS-to-DB2 propagation, this is the format of the segment as it appears *after* processing by the exit.

The values you can specify are:

F Database segment of fixed length

V Database segment of variable length

MAXNBR=1 (REQUIRED if segment is an extension segment of a mapping case 2 propagation request)

Is included if the IMS segment can have no more than one occurrence within an occurrence of its parent IMS segment.

FREQ= *n* **(OPTIONAL)**

The FREQ keyword has no effect on the extract and/or propagation of a propagation request belonging to a generalized mapping case.

Refer to the appropriate DataRefresher or DXT documentation for a complete description of FREQ keyword.

DESC=’ *comment* **’ (OPTIONAL)**

Saves a comment in the DataRefresher FDTLIB about the segment you are describing. For more information about coding this keyword, see Appendix A, “Coding the Description ,” on page 591.

If the IMS segment needs to be processed by a IMS DPROP Segment exit routine, provide either a DATAEXIT= or EXIT= keyword, and an XBYTES= keyword.

DATAEXIT= *exitname* / **EXIT=***exitname* **(REQUIRED if segment is processed by a Segment exit routine)**

Is used if the segment within the PCB must be interpreted or changed by a Segment exit routine before DataRefresher can extract from it and before IMS

DPROP can propagate from it. Replace *exitname* with the name of the Segment exit routine that you wish to invoke.

If the IMS segment contains internal segments propagated by mapping case 3 propagation requests, you must often use a Segment exit routine to define the IMS segment. For example, propagating internal segments with TYPE=E propagation requests requires that the containing IMS segment be defined with a Segment exit routine.

For user mapping, IMS DPROP ignores the DATAEXIT keyword and does not call Segment exit routines during propagation. (The exit identified on DATAEXIT is called by DataRefresher only during the extract.)

If you specify DATAEXIT or EXIT you must also specify XBYTES.

XBYTES= *n* **(REQUIRED if segment is processed by a Segment exit routine)**
Is used if the segment within the PCB must be interpreted or changed by a Segment exit routine before DataRefresher can extract it and before IMS DPROP can propagate from it. Replace *n* with the length in bytes of the segment in its IMS DPROP format. If you are performing IMS-to-DB2 propagation, this is the format *after* the Segment exit routine processes the segment.

n can be any integer from 1 through 32760.

For variable-length segments, specify the maximum length in bytes of the segment after it is processed by the Segment exit routine. Be sure to add 2 to the XBYTES value for the 2-byte length field located at the start of the segment.

If you specify XBYTES, you must also specify DATAEXIT or EXIT.

EODCALL=N | Y (OPTIONAL and only used when DATAEXIT or EXIT is specified)
Requests that DataRefresher make an end-of-data (EOD) call to the Segment exit routine during the extract. IMS DPROP does *not* issue EOD calls.

EOD calls, if requested, are made in addition to the normal calls for segment occurrences, and are useful if the exit is summarizing.

For root-level segments from an IMS database, EOD means at the end of the data set or database, subject to any search strategy.

For lower level segments from an IMS database, EOD is relative to each occurrence of the parent. For each occurrence of the immediate parent segment, an EOD call is made if requested.

SEGMENT Statement for Internal Segments

The SEGMENT statements define both IMS segments and internal segments. They describe to DataRefresher the hierarchical structure of the IMS segments and the internal segments.

Write the SEGMENT statements in hierarchical order (from top to bottom and left to right). You can provide up to 255 SEGMENT statements for each DXTPCB. For generalized mapping, you must include:

- A SEGMENT statement for each internal segment type propagated by mapping case 3 propagation requests
- A SEGMENT statement for each propagated segment and for the segment's physical parents and ancestors (up to the root segment)

For a description of the SEGMENT statement for IMS segments, refer to "SEGMENT Statement for IMS Segments " on page 32.

You can use any of the following forms of the START= keyword on the SEGMENT statement for internal segments:

START=*n*

Use START=*n* when the first occurrence of the internal segment has a fixed start position within its containing IMS segment. *n* identifies the fixed start position.

START=*fieldname+n*

Use START=*fieldname+n* when the first occurrence of the internal segment starts *n* bytes from the end of the named field. The named field must be a field located in the containing IMS segment.

If the named field has a variable length, then the first internal segment has a variable start position.

START=*segname+n*

Use START=*segname+n* when the IMS segment contains multiple internal segment types. When specifying START=*segname+n*, the first occurrence of the internal segment starts *n* bytes from the last occurrence of the other internal segment identified by segname.

Parameter Descriptions for the SEGMENT Statement for Internal Segments:

SEGMENT (REQUIRED)

Provide one SEGMENT statement for each internal segment propagated by a mapping case 3 propagation request.

The SEGMENT statements should describe both the IMS segments and the internal segments in hierarchical order (from top to bottom and left to right).

Internal segments can be fixed or variable in length and can contain fixed-length and variable-length fields. The variable-length fields can be either variable-length character (VC) or variable-length graphic (VG) data types.

NAME= *segname* **(REQUIRED)**

Names the internal segment.

The segment name must be unique to the DXTPCB. It must be different from all other segment names in the DXTPCB. In contrast to DataRefresher, IMS DPROP permits only names of up to 8 characters. The name can contain alphanumeric or special characters.

PARENT= *psegname* **(REQUIRED)**

Names the containing IMS segment.

If you are using a Segment exit routine, define it on the DATAEXIT= keyword of the SEGMENT statement describing the containing IMS segment *psegname* (*not* on the SEGMENT statement describing the internal segment).

FORMAT= FI | VI (REQUIRED for internal segments)

Indicates the format of the internal segment.

The values you can specify are:

FI Internal segment of fixed length

VI Internal segment of variable length

For variable-length internal segments (as opposed to variable-length IMS segments), IMS DPROP does not require that the segment start with a two-byte length field. See the description of the NEXT= keyword on page 37 for information on how IMS DPROP determines the length of a variable-length internal segment.

OCCURS= *n* | *fieldname* **(REQUIRED for internal segments)**
Tells IMS DPROP how many occurrences of this internal segment type there are in the containing IMS segment. OCCURS can be an integer or the name of counter field in the containing IMS segment, whose value is the number of occurrences.

n For a fixed number of occurrences, replace *n* with that number.

fieldname
For a variable number of occurrences, replace *fieldname* with the name of the field that contains the value for the number of occurrences. This count field must be located in the containing segment and must physically precede the start of the internal segment.

For TYPE=E propagation requests, you cannot propagate the count field.

The count field must have a numeric data type and must have a scale of zero. It cannot be a floating point number.

When propagating an internal segment with a TYPE=E propagation request, IMS DPROP requires that the internal segment be defined as having a variable number of occurrences and that you specify OCCURS=*fieldname*.

START= *n* | *fieldname+n* | *segname+n* **(REQUIRED for internal segments)**
Specifies the starting position of the first occurrence of the internal segment type within the containing IMS segment.

If the containing IMS segment is processed by a Segment exit routine, your definition of the start position should apply to the containing IMS segment as described to IMS DPROP, that is, for IMS-to-DB2 propagation, the description applies to the segment format *after* processing by the Segment exit routine.

For a fixed starting position, replace *n* with the starting position of the first occurrence of the internal segment type within the containing IMS segment. The value *n* must be a positive integer. The first byte within a segment is considered to have the start position of 1, not 0. If the containing IMS segment has a variable-length format, be sure to account for the 2-byte length field at the start of the segment. For example, a value of START=1 means that the internal segment begins at the start of the length field of the containing IMS segment, not at the byte following the length field.

For a variable starting position, write one of the following:

fieldname+n
Replace *fieldname* with the name of a field that is defined in the containing IMS segment.

Replace *n* with the number of bytes from the end of the named field to the starting position of the first occurrence of this internal segment. *n* must be a positive integer.

If the named field and this internal segment type are adjacent, write *fieldname*+1.

If the named field has a variable length, then the first occurrence of the internal segment type has a variable start position.

segname+n
Use *segname+n* if the containing IMS segment contains multiple internal segment types and you currently describe the *m*th (*m*>1) internal segment type.

Replace *segname* with the name of another internal segment type defined in the same containing IMS segment. Be sure to define this other internal segment type with a SEGMENT statement.

Replace *n* with the number of bytes from the end of the last occurrence of the named segment to the starting position of the first occurrence of this internal segment type. *n* must be a positive integer.

If the two internal segment types are adjacent, write *segname*+1.

BYTES= *n* **(OPTIONAL for fixed-length internal segments)**
For fixed-length internal segments, *n* specifies the length in bytes of the internal segment.

You can specify either the BYTES= keyword or the NEXT= keyword, but not both.

NEXT= *fieldname+n* **(REQUIRED for variable-length internal segments; OPTIONAL for fixed-length internal segments)**
Tells DataRefresher and IMS DPROP how to step from one occurrence of the internal segment to the next.

For fixed-length internal segments, you can specify either the BYTES= keyword or the NEXT= keyword, but not both.

Write NEXT=*fieldname+n* as follows:

- Replace *fieldname* with the name of a field located in the internal segment.
- Replace *n* with the number of bytes from the end of the named field to the next occurrence of the internal segment. *n* must be a positive integer.

  If the named field and the next occurrence of the internal segment are adjacent, write *fieldname*+1.

If the named field has a variable length, then the internal segment has a variable length.

For variable-length segments, IMS DPROP expects that each occurrence of a variable-length internal segment be large enough to contain the position identified by (*fieldname + n*) - 1.

FREQ= *n* **(OPTIONAL)**
The FREQ keyword has no effect on the extract or propagation of a propagation request belonging to a generalized mapping cases.

Refer to the appropriate DataRefresher or DXT documentation for more information on the FREQ keyword.

DESC=’ *comment* **’ (OPTIONAL)**
Saves a comment in the DataRefresher FDTLIB about the segment you are describing. For more information about coding this keyword, see Appendix A, “Coding the Description ,” on page 591.

FIELD Statement

FIELD (REQUIRED)
Provide one FIELD statement for each field that you want to describe to DataRefresher and to IMS DPROP. You must provide at least one FIELD statement for each propagated field. In addition, you need to provide a FIELD statement for the length field of a variable-length character (VC) field and for a variable-length graphic (VG) field.

When describing an IMS segment containing internal segments, remember that you might need to provide FIELD statements for the count field and for the field used to identify the start of the internal segment. These fields are referenced by the OCCURS=*fieldname* and the START=*fieldname+n* keywords of the internal segment.

If you are performing DB2-to-IMS synchronous propagation and using DXT Version 2 Release 5 or DataRefresher, consider using FIELD statements to define portions of a segment that are not propagated. During DB2-to-IMS synchronous propagation, when inserting a segment or extending a variable-length segment, IMS DPROP initializes nonpropagated fields that are included in the DXTVIEW with the default value associated with the field data type. Space that is not defined with FIELD statements and fields that are not included in the DXTVIEW are instead initialized with binary zeroes. For propagation requests created with older DXT releases, IMS DPROP initializes all nonpropagated fields with binary zeroes.

When describing segments containing redefined data, you can use a FIELD statement to define each field, even fields describing redefined data. You define which particular segment redefinition and fields are propagated by a particular propagation request on the CREATE DXTVIEW command, the SUBMIT/EXTRACT command, or both.

The FIELD statements for a segment can appear in any order. Include at least one FIELD statement for each PCB you describe. You need not include a FIELD statement for every described segment.

DataRefresher distinguishes between:

- *ordinary* fields, and
- *sequence* fields. DataRefresher distinguishes between two types of sequence fields: the *key field* of the IMS segment and *index fields*. This information does not describe index fields in detail because they are not relevant to a propagation request belonging to generalized mapping cases. See the appropriate DataRefresher or DXT documentation for information on index fields.

  With the exception of root segments, DataRefresher does not require that you identify sequence fields. IMS DPROP accesses the DBDLIB for information about key fields; therefore, for non-root segments, you do not have to explicitly describe the key field as a sequence field.

  DXT Version 2 Release 5 and DataRefresher support the following data types for sequence fields:

  C Character
  Z Zoned decimal
  P Packed decimal
  H Halfword binary
  F Fullword binary

  If this is not appropriate for data propagation (for example, the field needs to be propagated to a graphic DB2 column), you can do either of the following:

  – Choose not to define the IMS key field to DataRefresher as a sequence field, for non-root segments.
  – Define the IMS key field twice with two different DataRefresher FIELD statements (and two different names); one identifies the field as a sequence field, and the other identifies the field as an ordinary, nonsequence field. The DataRefresher EXTRACT statement refers to the ordinary field.

For an extended-function propagation request (PRTYPE=E), the fully concatenated key of an entity segment must be totally mapped. (PRTYPEs are discussed on page 51). You can do this by defining the IMS key field of the entity segments and the IMS key fields of their physical parents/ancestors with FIELD statements. To define the IMS key field of a segment, you can either:

– Define the key field as one field with one FIELD statement, or
– Use multiple FIELD statements to define subfields that completely overlay the IMS key field.

FIELD can describe fixed- and variable-length fields. Variable-length fields can be of either variable-length character or variable-length graphic data types. You tell DataRefresher the length of the variable-length field by specifying a length field with LFIELD. (See LFIELD on page 43 for more information on length fields.) Do not define a variable-length field as a sequence field.

If you use a Segment exit routine to preprocess records in your source data, use FIELD to describe the segments' fields as they will appear *after* processing by the exit.

IMS DPROP and DataRefresher require that the Segment exit routine not change the value, length, or start position of the following types of fields:

- The IMS key field of a database segment
- Fields defined as sequence fields
- Fields mapping to a column of the primary DB2 key

If your Segment exit routine does not observe these rules, DataRefresher DEM and propagation failures can result.

NAME= *fieldname* **(REQUIRED)**
Specifies the name of the field being described. For ordinary fields, replace *fieldname* with the name of the field being described. You can also make up your own field names for ordinary fields.

For sequence fields, replace *fieldname* with the field name defined to IMS in the DBD. For a key field, this is the name defined on a FIELD statement of the IMS DBD; for an index field, this is the name defined on an XDFLD statement of the IMS DBD.

For ID fields propagated by TYPE=E propagation requests to the DB2 primary key, IMS DPROP requires that the ID field be defined in the IMS DBD, except for ID fields located in an internal segment. For such ID fields, replace *fieldname* with the field name defined to IMS in the DBD.

Field names must be valid DataRefresher, DataRefresher-quoted, or double-byte character set (DBCS) names. If the field name is a DBCS name, the name must contain *only* DBCS characters. No single-byte characters are permitted. The name must be bracketed by a shift-out and a shift-in character.

Sequence field names must be eight characters or less. Ordinary field names can be up to 32 characters including the shift-out and shift-in characters and excluding double quotes.

By assigning unique field names across all segments, you can avoid having to provide field names on the DXTVIEW command.

TYPE= **C** | *value* **(OPTIONAL)**
Indicates the type of data in the field.

The values you can specify are:

A Date data
B Single-byte binary data
C EBCDIC character string
D Long (8-byte) floating-point number
E Short (4-byte) floating-point number
F 32-bit, signed binary integer
G Only DBCS data
H 16-bit, signed binary integer
P Packed decimal number
S Time stamp data
T Time data
VC Variable-length character data
VG Variable-length graphic data
xx User-defined data type; *xx* is replaced with the name of the user data type that you defined on a CREATE DATATYPE command
Z Zoned decimal number

Refer to the appropriate Administrators Guide for your propagation mode for a detailed description of the above data formats. The Administrators Guide appropriate for your propagation mode also describes data conversions between IMS fields and DB2 columns that are supported by IMS DPROP.

DataRefresher always interprets sequence fields as type C. If this is not appropriate for data propagation (for example, the field needs to be propagated to a numeric DB2 column), you can:

- Choose not to define the IMS key field to DataRefresher as a sequence field, for non-root segments, or
- Define the IMS key field twice with two DataRefresher FIELD statements and two different names; one identifies the field as a sequence field, and the other identifies the field as an ordinary, nonsequence field. The DataRefresher EXTRACT statement refers to the ordinary field.

For IMS fields and IMS-to-DB2 propagation, IMS DPROP supports all of the following formats for date and time data: ISO, USA, EUR, or JIS. Support of IMS fields with other date and time formats requires a Field exit routine.

For DB2-to-IMS synchronous propagation, IMS DPROP maps date and time data to only *one* of the following formats: ISO, USA, EUR, or JIS. During IMS DPROP customization, the system administrator indicates which format will be supported on the DATE and TIME keywords of the EKYGSYS macro of DPROPGEN. Support of the other date and time formats requires a Field exit routine.

When you define a field during DBD generation, you must specify an IMS data type on the FIELD control statement. This can be either:

C Alphanumeric
F 32-bit signed binary integer
H 16-bit signed binary integer
P Packed decimal
X Hexadecimal data

When specifying TYPE for a DBD field, do not assume that the IMS field type determines the DataRefresher field type. Ordinarily, it does for fields of types P, F, or H, but does not always for fields of types C or X. The C or X designation

traditionally has been used for fields whose data types are foreign to IMS. Consult the database administrator for information about how the fields actually represent the data.

When processing a FIELD statement for a user-defined data type, DataRefresher retrieves the definition of your data type from the CREATE DATATYPE specification and issues a DEF call to the Field exit routine associated with the data type. Therefore, before providing a FIELD statement with a user data type, you must define the data type with a CREATE DATATYPE statement and provide a Field exit routine.

For a propagation request belonging to a user mapping case, Field exit routines are not called by IMS DPROP during data propagation.

START= *n* **(REQUIRED)**

Specifies the starting position of a field in a segment. The first byte within the segment is considered to have the start position of 1, not 0.

Replace *n* with a value indicating where the field begins in its segment. Unless the segment is processed by a Segment exit routine, use the following segment format to determine the starting position of the field: the segment format that IMS provides after it retrieves the segment with a DL/I call using a PCB that references the physical DBD without field sensitivity. The value of *n* can come from the START keyword of the DBDGEN FIELD statement defining the field.

For a logical child segment type, the segment always begins in the DL/I call I/O area with the fully concatenated key of the logical parent. This is true even if you define the fully concatenated key of the logical parent in the DBD as virtual. Therefore, a field starting in position 1 starts in the fully concatenated key of the logical parent.

If you are describing an IMS segment with a variable-length format, be sure to account for the 2-byte length field at the start of the segment. For example, a value of START=1 means it begins at the start of the length field, not at the byte following the length field.

Variable-length *internal* segments are not required to start with a 2-byte length field.

You cannot propagate segments containing fields with variable start positions with the generalized mapping logic of IMS DPROP. If you need to propagate fields with variable starting positions, you can either:

- Provide a Segment exit routine to edit the segment format so each field has a fixed starting position. On the START keyword, define the fixed starting position of the field in the reformatted segment format.
- Perform your mapping and propagation with a Propagation exit routine. On the START keyword, define a variable starting position. Refer to the appropriate DataRefresher or DXT documentation for more information on specifying variable starting positions.

BYTES= *n* **(REQUIRED for sequence fields and ordinary fields with lengths not determined by data type, namely ordinary fields of type A, C, P, S, T, VC, VG, and Z)**

Specifies how many bytes a field occupies or, for a variable-length field (VC or VG data types), the maximum number of bytes a field can occupy.

For ordinary fields, replace *n* with the field length in bytes.

For sequence fields, replace *n* with the field length in bytes. For index fields, this is the sum of the lengths of the component fields in the source segment.

For fixed-length segments, IMS DPROP requires that propagated fields be totally contained in the segment; the start and end of the field must be contained in the segment. For variable-length segments, each field must be either:

- Totally outside of the particular segment occurrence; the field start position must be beyond the end of the particular segment occurrence, or
- Totally contained within the segment.

If you are writing a DXTPSB description that describes a user-defined data type, the value you code here depends on the value you coded on the SRCBYTES keyword of the CREATE DATATYPE command. If SRCBYTES has a fixed number on the CREATE DATATYPE command, you do not need to include the BYTES keyword on the CREATE DXTPSB command. If, however, VARIES was specified for SRCBYTES on the CREATE DATATYPE command, then you must specify the length of this field with the BYTES keyword here.

If you specify BYTES for fields whose lengths are determined by data type, you must specify the correct length as shown in Table 3.

Table 3. Lengths Determined by Data Type

Data Type	Exact Length
B	1
D	8
E	4
F	4
H	2
xx	value of SRCBYTES specified on CREATE DATATYPE command

If you specify BYTES for fields whose lengths are not determined by data type, you must specify a length in the ranges shown in Table 4.

Table 4. Lengths Not Determined by Data Type

Data Type	Minimum Length	Maximum Length
S	19[1]	26
A	10[1]	10
C	1	254
G	2	254[2]
P	1	16
T	8[1]	8
VC	1	32767
VG	2	32766[3]
Z	1	16

[1]DataRefresher supports timestamp, date and time fields with smaller lengths; however, IMS DPROP requires that they have a length of 19, 10, and 8, respectively. [2]127 DBCS characters, BYTES must be even. [3]16383 DBCS characters, BYTES must be even.

Note: VG and G data types represent DBCS data; therefore, you must use two bytes to represent one DBCS character when you calculate the length or starting position. When defining the length of a VG and G field for DataRefresher and IMS DPROP, you must express the length in terms of bytes. However, when defining a graphic column to DB2, express its length in terms of the number of DBCS characters.

When defining the length of a packed decimal field for DataRefresher and IMS DPROP, you must express the length in terms of bytes. However, when defining a decimal column to DB2, express its length in terms of digits.

LFIELD= *fieldname* **(REQUIRED by IMS DPROP for variable-length fields)**
Is used only for ordinary fields to name the field that contains the length of a variable-length field (field type of VC or VG). The field named with LFIELD must physically precede the variable-length field within the IMS segment.

The value of the field named with LFIELD must be even if you define the field as a VG data type. The length field must be numeric data, and must have a zero scale if it is packed or zoned decimal; it cannot be floating point. A length field can have a zero value.

Notes®:

1. Typically, you will not propagate a length field; you will propagate only the VC/VG field. Therefore, do not include the name of length fields in the EXTRACT statement.
2. Do not specify an LFIELD keyword for fields that are transformed by a Field exit routine into VC/VG format. For such fields, the length must be provided by the Field exit routine.

Refer to Figure 26 on page 30 for an example of the LFIELD keyword.

SCALE= *n* | **0 (REQUIRED for ordinary fields of P and Z data types)**
Indicates the position of the decimal point in the decimal representation of the field type. Do not use SCALE for sequence fields, because all sequence fields are of type C or G.

If you are writing a DXTPSB description that describes a user-defined data type, the value you code here depends on the value you coded on the SRCSCALE keyword of the CREATE DATATYPE command.

- If SRCSCALE has a fixed number on the CREATE DATATYPE command, you do not need to include the SCALE keyword on the CREATE DXTPSB command. If you do include SCALE, it must be equal to the SRCSCALE keyword.
- If, however, VARIES was specified for SRCSCALE on the CREATE DATATYPE command, then you must specify the scale of this field with the SCALE keyword here.

The values you can specify are:

n If you want the decimal point to be placed to the left of the last *n* digits in the decimal representation, replace *n* with a positive integer.

The value of *n* can range from 1 through the number of decimal digits contained in the field being described. For an *x*-byte field of type Z, the maximum scale, *n*, is *x*. For an *x*-byte field of type P, the maximum scale, *n*, is *2x-1*.

0 If there is no decimal point in the number. The value of the field is assumed to be an integer.

CONV= *exitname* **(NOT SUPPORTED by generalized mapping logic of IMS DPROP)**

Specifies the name of a date/time conversion exit. Date/time conversion exits are supported by DataRefresher but are not supported by the generalized mapping logic of IMS DPROP. If you need to convert the format of your date/time fields, you can use Field exit routines instead of date/time conversion exits.

You cannot propagate segments containing fields with a date/time conversion exit with the generalized mapping logic of IMS DPROP.

SEQFLD=R | V (REQUIRED for sequence fields, not used for ordinary fields)

Specify SEQFLD=R for the key field of an IMS segment.

DataRefresher supports both the definition of SEQFLD=R and SEQFLD=V. Do not specify SEQFLD=V in DXTPCBs used to define a PR belonging to a generalized mapping case.

SEQUENCE=ASC (OPTIONAL for sequence fields, not used for ordinary fields)

Indicates that sequencing is by ascending value of the field.

UNIQUE= Y | N (OPTIONAL for sequence fields, not used for ordinary fields)

Indicates whether a sequence field has unique values.

The values you can specify are:

Y If the field has unique values. If you specify UNIQUE=Y (or if you default to UNIQUE=Y), during the extract, DataRefresher checks that the sequence field has unique values.

N If the field does not have unique values.

DESC=’ *comment* **’ (OPTIONAL)**

Saves a comment in the DataRefresher FDTLIB about the field you are describing. For more information about coding this keyword, see Appendix A, “Coding the Description ,” on page 591.

Using CREATE DXTPSB for User Mapping Cases

With the exception of the restrictions described below, you can use the full defining and mapping capabilities of DataRefresher for DXTPSBs to define a propagation request belonging to a user mapping case.

For a propagation request belonging to a user mapping case, your DataRefresher definitions are used for mapping performed by DataRefresher during the IMS data extract; however, the definitions are not used by IMS DPROP during data propagation. Instead, during propagation, the mapping is performed by the Propagation exit routine you identified on the EXITNAME keyword of the MAPUPARM keyword of the DataRefresher SUBMIT command.

For user mapping cases, Segment exit routines, Field exit routines, and Date/Time Conversion exit routines are only called by DataRefresher during the IMS data extract; they are not called by IMS DPROP during data propagation. You can also call these exit routines from your Propagation exit routine during data propagation.

Refer to the appropriate DataRefresher or DXT documentation for more information on:

- The full mapping capabilities of DataRefresher.
- How to define a DataRefresher mapping that is compatible with the mapping performed during propagation by your Propagation exit routine.

If you cannot define compatible DataRefresher mapping, then you need to provide a program for the IMS extract and DB2 load.

Some IMS DPROP requirements associated with using CREATE DXTPSB to define a propagation request belonging to a user mapping case are:

- A DXTPCB statement is required for each IMS database propagated by the Propagation exit routine.
- The DBD name specified in the DBNAME keyword of the DXTPCB statement must be the name of a physical IMS DBD.
- A DataRefresher SEGMENT statement is required for each segment propagated by the Propagation exit routine; that is, for each segment identified on the PROPSEGM keyword of the MAPUPARM keyword of the DataRefresher SUBMIT command.
- The segment names specified on the NAME keyword of the DataRefresher SEGMENT statement must be the same as the segment name defined in the physical IMS DBD identified on the DBNAME keyword.

CREATE DXTVIEW

Use CREATE DXTVIEW to create a unique data description, called a DXTVIEW, of an IMS database. You create a DXTVIEW after creating a DXTPSB but before defining a propagation request. This command allows you to describe views of the IMS data to be extracted and propagated. Each CREATE DXTVIEW includes fields of only one hierarchical path of an IMS database.

In addition to identifying a DXTPSB and DXTPCB, the CREATE DXTVIEW command identifies:

- A bottom segment; all segments in the path from the root to the segment you specify on the SEGMENT keyword are included in the DXTVIEW
- A minimum segment; you identify this on the MINSEGM keyword.

Some IMS DPROP recommendations to be aware of if you are using CREATE DXTVIEW statements and with propagation requests belonging to the following mapping case include:

Case	Considerations and Recommendations
1	You can define DXTVIEWs in two ways: You can provide either one DXTVIEW for each hierarchical path of your DXTPSB or one DXTVIEW per entity segment. The first way allows the same DXTVIEW to be shared by multiple propagation requests, thereby reducing the number of DXTVIEWs that you need to define. • If you are providing one DXTVIEW for each hierarchical path of your DXTPSB, identify the lowest segment of the hierarchical path on the SEGMENT keyword. When you use this method, IMS DPROP determines the entity segment during processing of the EXTRACT statement. On the MINSEGM keyword, which is optional, specify the lowest segment of the hierarchical path. You can also specify a segment at the same or deeper hierarchical level than the entity segment. • If you are providing one DXTVIEW for each entity segment type, identify the entity segment on the SEGMENT keyword. On the MINSEGM keyword, which is optional, identify the entity segment.
2	Provide one DXTVIEW for each extension segment type. Identify the

extension segment on the SEGMENT keyword. On the MINSEGM keyword, which is mandatory, specify the name of the entity segment.

3 Provide one DXTVIEW for each internal segment type. Identify the entity/internal segment on the SEGMENT keyword. On the MINSEGM keyword, which is optional, identify the internal segment.

You can share the use of these DXTVIEWs with mapping case 1 propagation requests propagating higher level entity segments.

User Each segment propagated by the propagation request must be in the path of one of the DXTVIEWs referred to by the DataRefresher EXTRACT statement. Depending on the type of DataRefresher mapping you request for the IMS extract, DataRefresher can have additional requirements. Specifying MINSEGM depends on the type of DataRefresher mapping you request for the IMS extract. Refer to the appropriate DataRefresher or DXT documentation for rules for specifying the MINSEGM keyword and for information on additional SEGMENT keyword requirements in user mapping cases.

Figure 27 shows the syntax for the CREATE DXTVIEW command.

```
►►──CREATE──DXTVIEW──NAME=viewname──,DXTPSB=psbname──DXTPCB=pcbname,──────────►

►─,SEGMEMT=bsegname──┬──────────────────────────┬──────────────────────────►
                     └─,MINSEGM=msegname────────┘

►─┬─FIELD=──┬──┬──*──────────────────────────────────────────────────────────►
  └─FIELDS=─┘  │  ┌──────────────────────────┐  └─,DESC='comment'──────────┘
               │  ▼                          │
               └──(──┬─fieldname──────────────┬──)──┘
                     └─alias=segname.fieldname┘

►─;──────────────────────────────────────────────────────────────────────────►◄
```

Figure 27. The CREATE DXTVIEW Command

CREATE DXTVIEW (REQUIRED)
Specifies that a DXTVIEW is being created.

NAME= *viewname* **(REQUIRED)**
Assigns a name to the DXTVIEW. Replace *viewname* with the name you assign. The name can be a DataRefresher name, DataRefresher quoted name, or a DBCS name. It must be unique among all the names for DXTVIEWs in your FDTLIB. DataRefresher supports up to 32 characters for DXTVIEW names.

DXTPSB= *psbname* **(REQUIRED)**
Creates a DXTVIEW of a database PCB. Replace *psbname* with the name of the DXTPSB description that contains the DXTPCB description that your DXTVIEW is based on.

DXTPCB= *pcbname* **(REQUIRED)**
Specifies the name of the DXTPCB. Replace *pcbname* with the name of the DXTPCB description that your DXTVIEW is based on.

SEGMENT= *bsegname* **(REQUIRED)**
Identifies the bottom segment for your DXTVIEW. All segments in the path, from

the root to the bottom segment you specify here, are automatically included in the DXTVIEW. Replace *bsegname* with the name of the bottom segment, which must be the name of a database segment or internal segment in the DXTPSB description. It need not, however, be the bottom segment of the hierarchy. If the *bsegname* is a segment from the IMS hierarchy, then no internal segments are available through the DXTVIEW, even though they can physically reside in *bsegname*.

If you are specifying SEGMENT for a DXTVIEW used by a propagation request belonging to the following mapping cases, the indicated rules apply:

Case	Rules
1	Identify either of the following on the SEGMENT keyword: • The entity segment • A segment in the same hierarchical path as the entity segment located at a deeper level than the entity segment
2	Identify the extension segment on the SEGMENT keyword. If a propagation request has multiple extension segments, provide one DXTVIEW for each extension segment, and identify the extension segment on the SEGMENT keyword.
3	Identify the entity segment (the internal segment) on the SEGMENT keyword.
User	Each segment propagated by the PR must be in the path of one of the DXTVIEWs referenced by the DataRefresher EXTRACT statement. Depending on the type of DataRefresher mapping you request for the IMS extract, DataRefresher can have additional requirements. Refer to the appropriate DataRefresher or DXT documentation for information on rules for specifying the SEGMENT keyword.

MINSEGM= *msegname* **(REQUIRED for some mapping cases)**
Identifies the minimum required segment that must be present in your DXTVIEW path. You can specify MINSEGM for any segment from the path over which you define DXTVIEW, including any internal segment.

Replace *msegname* with the name of the segment representing the minimum requirement for data extraction qualification. If MINSEGM is not specified, the default is the segment specified with SEGMENT.

IMS DPROP has the following rules for specifying MINSEGM: If you are specifying MINSEGM for a DXTVIEW used by a propagation request belonging to the following mapping cases, the indicated rules apply:

Case	Rules
1	The MINSEGM keyword is optional. If you specify MINSEGM, the identified *msegname* must be in the same hierarchical path as the segment identified on the SEGMENT keyword, must not be at a lower hierarchical level than *bsegname*, and must not be at a higher hierarchical level than the entity segment.
2	Identify the entity segment on the MINSEGM keyword.
3	The MINSEGM keyword is optional. If you specify MINSEGM, the identified *msegname* must be the entity segment (the internal segment).
User	The MINSEGM keyword depends on the type of DataRefresher

mapping you request for the IMS extract. Refer to the appropriate DataRefresher or DXT documentation for information on rules for specifying the MINSEGM keyword.

FIELD= | FIELDS= (REQUIRED)

Specifies the fields you want in the DXTVIEW. You must specify at least each field that you want to extract and propagate.

You can name only fields that are described in the DXTPCB description and that lie in the path of the DXTVIEW determined by the bottom segment, which is designated by the SEGMENT keyword.

You can specify one of the following values for FIELD | FIELDS:

- * (asterisk)
- *fieldname*
- *alias=segname.fieldname*
- A combination of both *fieldname* and *alias=segname.fieldname*

If you specify more than one *fieldname* or *alias=segname.fieldname*, then you must enclose the list in parentheses and separate the fields with commas.

*** (asterisk)**

Makes visible every field in the path in your DXTPCB description.

When specifying fields for a DXTVIEW of a database, you cannot use FIELD=* if the path for the view contains two or more eligible fields that have the same names. Use aliases to distinguish fields with the same name.

You can use FIELD=* when fields in the DXTPCB description have duplicate names, as long as the fields in the path for the DXTVIEW being defined do not contain these duplicate names.

fieldname

Is used to identify the fields that you want to view. Replace *fieldname* with a list of field names, separated by commas. Field names must be DataRefresher, DataRefresher-quoted, or DBCS names.

Replace *fieldname* with the name of a field if no other fields with the same name exist. For a database description having duplicate field names, use alias=*segname.fieldname*.

alias= *segname.fieldname*

Assigns an *alias* to any field in your list. An alias is an alternative name for a field and must be a DataRefresher, DataRefresher-quoted, or DBCS name. That alias must then be used instead of the field name when the field is selected through your DXTVIEW. A DXTPCB description can describe two or more fields with the same name, as long as these fields are in different segments of the database. If this is the case and you want these fields in your DXTVIEW, you must assign aliases to all but one of them. For example, if there are three fields with the same name in your DXTVIEW, you must assign aliases to two of them so that they can be distinguished from one another.

Note: The underlying DXTPCB description for a DXTVIEW can be updated. If it is, the DXTVIEW then refers to the fields in the updated DXTPCB description. Be sure when an update occurs that the dependent DXTVIEWs are still correct, or regenerate the DXTVIEWs to be compatible with the updated data description.

DESC=’ *comment* **’ (OPTIONAL)**
Saves a comment in the DataRefresher FDTLIB about the data description you are creating. For more information about coding this keyword, see Appendix A, “Coding the Description ,” on page 591.

SUBMIT/EXTRACT

This topic describes the DataRefresher SUBMIT command and the DataRefresher EXTRACT statement as they are used with IMS DPROP. Refer to the appropriate DataRefresher or DXT documentation for a complete description of the SUBMIT command and the EXTRACT statement.

The syntax for the SUBMIT command is shown in Figure 28. The syntax for the EXTRACT command is shown in Figure 30 on page 62.

```
►►──SUBMIT──EXTID=prid──,MAPEXIT=EKYMCE00──┬──────────────────────────────┬──►
                                           └─,MAPUPARM='prop-parms'─┘

►──┬──┬──────────────────────────────────────────────────────────────────────┬──►◄
   │  ▼  ┌─DB2─┐
   │  ,DBS=──┴─DSI─┘
   │  ,FORMAT=──┬─EBCDIC─┬──────────────────────────────────────────────────┤
   │            └─SOURCE─┘
   │  ,USERID=userid──────────────────────────────────────────────────────────┤
   │  ,NODE=nodeid────────────────────────────────────────────────────────────┤
   │  ,MSGS(jcsddname,msgtable)───────────────────────────────────────────────┤
   │  ,JCS=jcsddname──────────────────────────────────────────────────────────┤
   │  ,EXTDATA=──┬─ddname──────────────────────────────────────────────────┤
   │             └─(dsn,unit,vol,blk,pass)─┘
   │  ,CD=──┬─JCS─────────────────────────────────────────────────────────┤
   │        └─EXTDATA─┘
   │  ,DECIMAL=──period───────────────────────────────────────────────────────┤
   │  ,USERDECK=──'optionlist'────────────────────────────────────────────────┤
   │  ,ACCOUNT=──'actinfo'────────────────────────────────────────────────────┤
   │  ,NULLS=──┬─Y─┬──────────────────────────────────────────────────────┤
   │           └─N─┘
   │  ,TYPE=──┬─NORMAL─┬──────────────────────────────────────────────────┤
   │          └─PERSIST─┘

►►──EXTRACT1──statement──;──────────────────────────────────────────────────►◄
```

[1]Refer to “EXTRACT ” on page 61 for a description of the EXTRACT statement.

Figure 28. The SUBMIT Command (Part 1 of 2)

where ’prop-parms’ are:

```
   ┌─,──────────────────────────────────────────────┐
   ▼                                                 │
►►─┬─PRTYPE=──┬─E─┬──────────────────────────────────┬──────────────────────────►◄
   │          ├─L─┤                                  │
   │          ├─F─┤                                  │
   │          └─U─┘                                  │
   ├─MAPCASE=─┬─1─┬──────────────────────────────────┤
   │          ├─2─┤                                  │
   │          └─3─┘                                  │
   ├─PATH=────┬─ID────────────────────────────────────┤
   │          └─DENORM────────────────────────────────┘
   ├─MAPDIR=──┬─HR─┬──────────────────────────────────┤
   │          ├─TW─┤                                  │
   │          └─RH─┘                                  │
   ├─TABQUAL2=qualifier──────────────────────────────┤
   ├─ERROPT=──┬─BACKOUT──────────────────────────────┤
   │          └─IGNORE────────────────────────────────┘
   ├─MAXERROR=number─────────────────────────────────┤
   │          ┌─ADD──┐
   ├─ACTION=──┤      │──────────────────────────────┤
   │          └─REPL─┘
   │          ┌─BUILDRUN──┐
   ├─PERFORM=─┤─BUILDONLY─┤──────────────────────────┤
   │          └─RUNONLY───┘
   ├─PRSET=setname───────────────────────────────────┤
   │          ┌─N─┐
   ├─PROPSUP=─┤   │──────────────────────────────────┤
   │          └─Y─┘
   ├─EXITNAME=exitname───────────────────────────────┤
   ├─PROPSEGM=(dbd/seg,...)──────────────────────────┤
   │          ┌─ASC─┐
   ├─KEYORDER=┤─DSC─┤──────────────────────────────┤
   │          └─ANY─┘
   ├─PCBLABEL=pcblabel───────────────────────────────┤
   ├─BIND=(bind options)─────────────────────────────┤
   ├─AVU=─────┬─Y─┬──────────────────────────────────┤
   │          └─N─┘
   │          ┌─Y─┐
   ├─DEFVEXT=─┤   │──────────────────────────────────┤
   │          └─N─┘
   └─COMMENT=(comments)──────────────────────────────┘
```

Figure 28. The SUBMIT Command (Part 2 of 2)

SUBMIT (REQUIRED)
Submits a DEM extract to DataRefresher.

EXTID= *prid* **(REQUIRED)**
prid identifies your PR/ER.

Replace *prid* with an identifier you want to use for this propagation request.

The propagation request ID must be a valid load module name. It cannot be the same name as other load modules in your installation. IMS DPROP generates an SQL update module for each propagation request belonging to a generalized mapping case, except for propagation requests defined with a mapping direction of RH; the SQL update module uses the propagation request ID as a load module name.

MAPEXIT=EKYMCE00 (REQUIRED by IMS DPROP)
Identifies the name of a DataRefresher Map Capture exit routine. When defining a IMS DPROP propagation request, you should specify the name of the IMS DPROP-provided Map Capture exit routine, EKYMCE00.

MAPUPARM=‘ *prop-parms* **’ (REQUIRED unless //MVGPARM provides a default value)**
Specifies the parameters that will be passed to the exit identified on the MAPEXIT keyword. These parameters must be enclosed in single quotes and must follow the rules of a DataRefresher literal.

MVG accesses the //MVGPARM data set for default propagation parameters. For more information on //MVGPARM, refer to Chapter 5, “//MVGPARM Data Set Propagation Parameters,” on page 133.

PRTYPE=E | L | F | U (REQUIRED unless //MVGPARM provides a default value)
Identifies the type of propagation request you are creating. Three propagation request types belong to the generalized mapping cases: extended function (E), limited function (L) and full function (F). A propagation request type of U, indicating user mapping, is available if you require mapping other than that provided in generalized mapping cases 1, 2 and 3. The majority of IMS segments in an installation can be propagated using the generalized mapping logic of PR types E, L and F. Because E supports more functions, including DB2-to-IMS synchronous propagation, it is preferred to L and F.

As an alternative to specifying PRTYPE=E, L, F or U, you can specify PRTYPE=EXTENDED, LIMITED, FULL or USER.

Refer to the appropriate Administrators Guide for your propagation mode for more information on the propagation request types.

E/EXTENDED
An extended-function propagation request belongs to generalized mapping case 1, 2 or 3 and supports hierarchical-to-relational propagation, relational-to-hierarchical synchronous propagation, and two-way synchronous propagation. Some characteristics associated with an extended-function PR include:

- A strong set of rules for the mapping of keys. These rules are validated in part by MVG. Any violation of these rules can result in propagation failures.
- Support for the CCU.
- Support for the DLU.

L/LIMITED
A limited-function propagation request belongs to generalized mapping case 1, 2 or 3 and supports only hierarchical-to-relational propagation. Some characteristics associated with a limited-function propagation request include:

- A weaker set of rules for the mapping of keys.
- Supports for the CCU.
- No support for the DLU.

F/FULL
A full-function propagation request has the same characteristics as a limited-function propagation request.

U/USER

U indicates user mapping. If you are providing your own mapping case, you must specify PRTYPE=U.

Some characteristics associated with user mapping include:

- It does not use the generalized mapping logic of IMS DPROP. Instead, you must provide your own Propagation exit routine.
- The RUP, and for synchronous propagation the HUP, are still invoked to provide certain functions:
  - Tracing
  - Checking propagation status
  - providing error logic.
- You are responsible for ensuring that the mapping/conversions performed during the extract by DataRefresher, based on your DataRefresher definitions, are compatible with the mapping/conversions performed during propagation by your Propagation exit routine.
- IMS DPROP does not call Segment exit routines and Field exit routines.
- IMS DPROP does not impose rules for the mapping of keys.
- No support for the CCU.
- No support for the DLU.

MAPCASE=1 | 2 | 3 (REQUIRED for PRTYPE=E, F and L unless //MVGPARM provides a default value)

Identifies the mapping case to which the propagation request belongs. If you selected a PRTYPE of U, this keyword is ignored, but if specified, it must contain a valid value.

1 identifies generalized mapping case 1. Mapping case 1 provides mapping between IMS data of one IMS segment type (“entity segment”) and its concatenated key to/from one relational table.

2 identifies generalized mapping case 2. Mapping case 2 provides mapping between IMS data of one IMS segment type (“entity segment”) and its concatenated key, and one or more immediately subordinated IMS segment types (“extension segments”) to/from one relational table. Each type of extension segment can occur once or not at all beneath the entity segment.

3 identifies generalized mapping case 3. Mapping case 3 provides mapping between IMS data of one internal segment (“entity segment”) and its concatenated key to/from one relational table.

PATH=ID|DENORM (REQUIRED for path data, unless //MVGPARM provides a default value)

Identifies the kind of non-key path data selected (from parent segments) in the propagation request.

ID Includes as path data only IMS ID fields, that is, fields that cannot change their value.

DENORM

Includes as path data fields that can change their value.

For propagation requests with PRTYPE=E, you cannot specify PATH=DENORM.

The default is no path data selected.

MAPDIR=HR | TW | RH (REQUIRED unless //MVGPARM provides a default value)

Indicates the direction in which the data is being propagated.

HR Propagation is to be performed in a hierarchical-to-relational direction only.

TW Synchronous propagation is to be performed both in a hierarchical-to-relational direction and in a relational-to-hierarchical direction.

RH Synchronous propagation is to be performed in a relational-to-hierarchical direction only.

Note: You cannot specify TW and RH for propagation requests defined with PRTYPE=L and PRTYPE=F.

For asynchronous or user asynchronous propagation you cannot provide propagation requests with a mapping direction of TW or RH.

TABQUAL2= *qualifier* **(OPTIONAL)**

If you specify an unqualified table name on the INTO clause of the EXTRACT statement for the propagated table, provide a TABQUAL2 keyword.

If you specify an unqualified table name on the INTO clause, IMS DPROP must find the description of a “model” table in the DB2 catalog. The model table can be (but does not have to be) the table being propagated. The model table should have the same attributes as the propagated table. IMS DPROP generates mapping information in the propagation request based on DB2 catalog descriptions of the model table.

You can specify the table name qualifier of the model table on the optional TABQUAL2 keyword (it must be a valid DB2 authorization ID). If you do not provide a TABQUAL2 keyword, then the user ID of the UIM job is used as the default table name qualifier of the model table.

The name of the model table is comprised of:

- The qualifier specified on the TABQUAL2 keyword (or the user ID of the UIM job)
- The unqualified table name specified on the INTO clause of the EXTRACT statement

If you specify an unqualified table name on the INTO clause of the EXTRACT statement, IMS DPROP assumes that the model table has been created and has the same attributes as the table propagated by the propagation request being defined.

ERROPT=BACKOUT | IGNORE (REQUIRED unless //MVGPARM provides a default value)

Indicates the error option. The values you can specify are:

BACKOUT

If propagation cannot be successfully completed, and you want IMS DPROP to back out all changes since the last commit point, select BACKOUT.

IGNORE

Helps to minimize operational risks associated with propagation for IMS and DB2 applications. If propagation cannot be successfully completed

(due to something other than unavailable DB2 resources) and you want IMS DPROP to ignore the error and return to its caller, select IGNORE.

When you select ERROPT=IGNORE, IMS DPROP does not abend the application; instead, it writes diagnosis information and returns to the application program without indicating errors. While this reduces failures that could impact existing applications, it results in inconsistencies between the IMS and DB2 data.

You might want to use ERROPT=IGNORE during the early phases of propagation in a production system when you are first learning how to define a propagation request. Later, you can use the SCU to change ERROPT=IGNORE to ERROPT=BACKOUT.

Do not use ERROPT=IGNORE as a replacement for the careful design, implementation, and test of data propagation.

MAXERROR= *n* **(OPTIONAL)**

If you select ERROPT=IGNORE, the MAXERROR value indicates how many propagation failures can be reported for this propagation request within a 15-minute interval. It also indicates how many errors will be documented with detailed information in the IMS DPROP trace. When this limit has been exceeded, propagation continues without error messages or detailed trace records being written.

Valid values for MAXERROR are between 0 and 32767, inclusive. The default value of 0 means *no limit*.

This keyword limits only those messages written to the OS/VS console and to the audit trail. It does not limit messages written to the optional //EKYPRINT data set and to the IMS DPROP trace.

ACTION= ADD | REPL (OPTIONAL)

Indicates whether you want to add a new propagation request or replace an existing propagation request.

ADD

When you select ADD, the IMS DPROP Map Capture exit, EKYMCE00, first checks to see that the propagation request does not already exist in the mapping tables. If it does not, EKYMCE00 creates a new propagation request in the IMS DPROP directory and the SQL module libraries. Otherwise, EKYMCE00 issues an error message and returns a nonzero return code to DataRefresher.

REPL

When you select REPL, EKYMCE00 either replaces the already existing propagation request, or, if the propagation request does not exist on the mapping tables, it adds the propagation request.

For synchronous propagation, if you replace an existing propagation request, the existing propagation request must be inactive or the IMS DPROP system must be emergency stopped.

PERFORM=BUILDRUN | BUILDONLY | RUNONLY (OPTIONAL)

Indicates whether you want to:

- Create a propagation request and run the data extraction (BUILDRUN).
- Create a propagation request without extracting any data (BUILDONLY).
- Run the data extraction only (RUNONLY).

The values you can specify are:

BUILDRUN
IMS DPROP creates a propagation request in the IMS DPROP directory, and DataRefresher UIM builds an ER. If no errors are encountered during the creation of the propagation request, DataRefresher stores the extract request definition in the EXTLIB for the subsequent data extraction by DataRefresher DEM.

BUILDONLY
Creates a propagation request in the IMS DPROP directory without extracting any data. BUILDONLY creates or replaces the propagation request in the IMS DPROP directory, but DataRefresher UIM does not build an ER. After the propagation request is generated, EKYMCE00 returns with a return code greater than 0, so DataRefresher does not extract the data.

For example, for synchronous propagation, you can code PERFORM=BUILDONLY for propagation requests having a mapping direction of RH. For such propagation requests, you will typically not extract the IMS data.

You can also use BUILDONLY when a DBD has been modified and receives a new version ID. If this modification does not affect the mapping (and therefore does not require another data extract), the database administrator can regenerate all propagation requests propagating this database without performing an extract.

RUNONLY
Extracts data without replacing the existing propagation request.

With this option, MVG checks that the PR already exists and that it exactly matches the new propagation request. If they match, and no other errors are encountered, DataRefresher stores the extract request definition into the EXTLIB for the subsequent data extraction. If they do not match, EKYMCE00 returns with a return code greater than 0, and DataRefresher does not extract the data. If you select RUNONLY, MVG requires that the propagation request be inactive, and ACTION=REPL.

PRSET= *setname* **(OPTIONAL)**
Identifies the set of propagation requests to which the propagation request belongs. If you do not provide a set name here, then the name defined at IMS DPROP generation will be used as a default.

Valid *setname* values are alphanumeric/special character names. Refer to the appropriate Administrators Guide for your propagation mode for more detail on the PRSET.

PROPSUP= N | Y (OPTIONAL)
Indicates whether propagation should be suppressed when a Segment exit routine issues a return code of 8.

N N is the default value for this parameter. Select N if you do not want to allow Segment exit routines to return a code of 8 during data propagation. When Segment exit routines return with a return code of 8, it is considered an error, and IMS DPROP issues an abend.

Y If you select Y and a Segment exit routine issues a return code of 8, it is not considered an error; instead, propagation for this propagation request is suppressed.

Suppression of propagation by a Segment exit routine results in inconsistencies between the IMS data copy and the DB2 data copy.

These inconsistencies can eventually result in propagation failures of other updates. Therefore, suppress propagation with caution.

If you choose Y and you do not specify the USE keyword of the CCU CHECK control statement, CCU executions could report inconsistencies where none likely exists.

EXITNAME= *exitname* **(REQUIRED for user mapping, unless //MVGPARM provides a default value)**

Identifies the Propagation exit routine performing propagation.

Do not specify this keyword for a propagation request belonging to a generalized mapping case.

PROPSEGM=(*dbd/seg,...* **) (REQUIRED for user mapping, unless //MVGPARM provides a default value)**

Identifies the IMS segments to be propagated by the PR being defined. Whenever one of the segments that you identified changes, the Propagation exit routine you specified on EXITNAME is invoked by the RUP to process the changed segments.

Specify your segments as follows:

```
(dbd/seg,seg,seg,...,dbd/seg,seg,seg,...)
```

Specify the database first and its segments next, followed by a second database, if any. Do not specify PROPSEGM for a propagation request belonging to a generalized mapping case.

If multiple propagation requests (with the same or different Propagation exit routine) identify the same segment on this keyword, the RUP issues multiple calls to the Propagation exit routines whenever that segment is changed.

This propagation parameter is used only by the RUP; that is, when the data is propagated from IMS to DB2. For DB2-to-IMS synchronous propagation, this parameter has no effect; the Propagation exit routine is called whenever the table specified on the INTO clause of the EXTRACT statement is updated.

KEYORDER=ASC|DSC|ANY (OPTIONAL)

Indicates the column sequence of the primary key index of the propagated table. The values you can specify are:

ASC All columns composing the primary key of the propagated table are in ascending sequence.

DSC All columns composing the primary key of the propagated table are in descending sequence.

ANY Indicates that MVG must access the DB2 catalog to get the defined ordering sequence of each column composing the primary key of the propagated table.

If you specify ANY, IMS DPROP's access to the DB2 catalog is typically lengthy and inefficient. Therefore, if you know that all primary key columns have been defined in the DB2 index as ASC or DSC, define them as ASC or DSC here.

If you incorrectly specify ASC or DSC, CCU executions can report inconsistencies that do not exist.

PCBLABEL=(*pcb name* **) (OPTIONAL)**
Identifies the name of the IMS PCB that the HUP uses to perform DB2-to-IMS synchronous propagation.

If this parameter is omitted and the propagation request is used to perform DB2-to-IMS synchronous propagation, then the name of the DXTPCB is used to identify the name of the IMS PCB that the HUP uses.

BIND=(*text* **) (OPTIONAL)**
Contains BIND options for the SQL update module.

If you want IMS DPROP to automatically bind the package of the SQL update module, you have to specify bind options, including the collection ID where this package is to be bound.

The bind options are provided in the form of a string of DB2 specifications, up to a maximum of 254 bytes. Enclose the DB2 specifications in parentheses, without a continuation character. An example of this is shown in Figure 29.

IMS DPROP ensures that the following keywords, which are not supported, are not specified in the string:
- MEMBER (IMS DPROP generates this operand)
- BIND
- PLAN
- COPY
- LIBRARY

If you do not specify the collection ID in the PACKAGE keyword, the other bind options are ignored and no package is created.

Recommendations for BIND Options:

- For performance reasons, specify VALIDATE(BIND) rather than the DB2 default of VALIDATE(RUN). This prevents DB2 from performing a package validation each time the RUP invokes an SQL update module.
- If you have specified an unqualified propagated table name in your propagation request, you should use the QUALIFIER operand to set the correct qualifier.

The default is no bind options.

Figure 29 shows an example of BIND options:

```
PACKAGE(ONLINE) OWNER(DBADMIN) QUALIFIER(DB2PROD) VALIDATE(BIND)
```

Figure 29. Example of BIND options for the SQL update module package

AVU=Y | N (OPTIONAL)
Avoids unnecessary updates. With this parameter, you can influence the performance of IMS-to-DB2 propagation. This parameter specifies whether or not IMS DPROP should determine (during a segment replace) whether at least one propagated field has changed.

Y The replace of an IMS segment results in a propagating SQL update, only if at least one propagated field has changed.

AVU=Y can reduce the total IMS DPROP path length when only a subset of the fields of a segment are propagated.

N The replace of an IMS segment results in a propagating SQL update, even if no propagated field has changed.

AVU=N can reduce the total IMS DPROP path length when, for example, all fields of a segment are propagated. When you specify AVU=N, IMS DPROP does not compare each field to determine if at least one of them has changed.

If you do not code AVU, IMS DPROP determines a default setting at propagation time, as follows:

- AVU=Y is the default value:
  - When propagating internal segments with mapping case 3 propagation requests, or
  - When propagating IMS segments with mapping case 1 and 2 propagation requests, and at least one non-key byte of the segment is not propagated.
- AVU=N is the default in all other cases.

DEFVEXT=Y | N (OPTIONAL)
Identifies how IMS DPROP handles extension segments propagated exclusively from default values and NULLs during DB2-to-IMS synchronous propagation.

This parameter is used only for mapping case 2 propagation requests performing DB2-to-IMS synchronous propagation. It tells IMS DPROP what to do with an extension segment during DB2-to-IMS synchronous propagation of an SQL INSERT or UPDATE:

- When all the target columns of an extension segment are either null or contain default values, and
- When the SQL insert/update changes at least one column mapped to the extension segment.

Y This is the default value. When DEFVEXT=Y, the propagation of the SQL update/insert results in zero occurrences of the extension segment type.

N When DEFVEXT=N, the propagation of the SQL update or insert results in an occurrence of the extension segment type, even if all of its fields are propagated from columns having a NULL or default value.

COMMENT=(*comments* **) (OPTIONAL)**
Specify up to 254 bytes of comments to document the propagation request. These comments are stored in the propagation request mapping table.

DBS=DB2 | DSI (OPTIONAL)
Identifies the type of table that your extract output is loaded into:

DB2 Loads your extract output into a normal DB2 table

DSI Loads your extract output into a DataPropagator Relational CCD table and prefixes the data with the four additional IBMSNAP columns that are required for the CCD table

Although DataRefresher supports additional values, IMS DPROP supports only DB2 and DSI. The default is DBS=DB2.

FORMAT=EBCDIC| SOURCE (OPTIONAL)
Specifies how DataRefresher formats the extracted data. The values you can specify are:

EBCDIC
Converts your extracted data to EBCDIC format.

SOURCE
Leaves the extracted data in source format.

DataRefresher reformats the source data (even when you specify FORMAT=SOURCE) when zoned decimal fields become packed decimal columns. If DataRefresher leaves your data in source format, it does not generate null indicators (hyphens) for invalid field values (except in the instance above in which DataRefresher does convert source data).

IMS DPROP has both requirements and recommendations for the value of the FORMAT keyword.

IMS DPROP Requirements

- FORMAT=SOURCE is required for a propagation request belonging to a generalized mapping case if the propagation request propagates floating point IMS fields.
- FORMAT=EBCDIC is required if your propagation request is propagating packed (P) or zoned (Z) IMS fields to DECIMAL DB2 columns with a different scale.

  FORMAT=EBCDIC is also required if your propagation request is propagating P or Z IMS fields with a nonzero scale to a SMALLINT/INTEGER DB2 column.

IMS DPROP Recommendation

Unless your propagation request is propagating P or Z IMS fields to decimal DB2 columns, you should specify FORMAT=SOURCE. This results in a smaller DataRefresher extract output file and uses fewer resources.

If you have a propagation request that propagates both floating-point data and P or Z IMS fields requiring a scale change, you should:

- Specify FORMAT=SOURCE, and
- Provide Field exit routines for those P or Z IMS fields requiring a scale change.

  The Field exit routines must convert the IMS field into a P or Z format having the same scale as the target DB2 table.

Note: If DataRefresher encounters a field that is improperly formatted after conversion, DataRefresher substitutes a hyphen in the blank preceding the field.

USERID= *userid* **(OPTIONAL)**
Identifies the user ID of your extract request.

Replace *userid* with an identifier of your choice.

While *userid* can be any (alphanumeric or special characters) name, you should replace it with the user ID assigned to you by your installation. This helps to identify who submitted the propagation request.

NODE= *nodeid* **(OPTIONAL)**
Identifies the node ID of your extract request.

Replace *nodeid* with an identifier of your choice.

While *nodeid* can be any (alphanumeric) name, you should replace it with the node ID assigned by your installation.

MSGS= (*jcsddname,msgtable* **) (OPTIONAL)**
Sends messages issued by DataRefresher about your SUBMIT command to a relational message table instead of the DXTPRINT data set. This keyword has no effect on IMS DPROP. Messages issued by IMS DPROP are always written to the MVGPRINT data set.

JCS= *jcsddname* **(OPTIONAL)**
Indicates that you want DataRefresher to generate an output job. In addition to

invoking the DB2 load utility, the job control statements (JCS) can call the SORT utility or any other program you want to use.

Replace *jcsddname* with the dd name of the input data set containing the JCS provided by you and associated with your extract request. The DataRefresher UIM stores your JCS and ER into EXTLIB. The DataRefresher DEM then writes the JCS to the //DXTOUTn data set during the extract. Also, the JCS that you provide must contain an *EO statement, which indicates where the DataRefresher DEM places the extracted data within your JCS. The *EO statement follows a //SYSREC DD * statement in your JCS.

If you do not include the JCS keyword, you must include the EXTDATA keyword.

EXTDATA= *ddname* **|** (*dsn,unit,vol,blk,pass*) **(OPTIONAL)**
Directs some or all of your DataRefresher extract output to a physical sequential data set (as opposed to an instream data set of the job submitted by DataRefresher based on the JCS= keyword).

If you do not include the EXTDATA keyword, you must write the JCS keyword.

This keyword has no effect on IMS DPROP.

For more information on this keyword, refer to the appropriate DataRefresher or DXT documentation.

CD=JCS | EXTDATA (OPTIONAL)
By providing this keyword, you indicate that you want DataRefresher to generate a load control deck or column descriptors for the DB2 load utility when it processes your extract request.

The value that you write determines where DataRefresher places the load control deck or column descriptors in the output.

If you do not provide a CD keyword, DataRefresher does not generate a load control deck. You must then provide a control deck.

JCS
DataRefresher places the load control deck in an instream data set of the job submitted by DataRefresher based on the JCS keyword specifications. Your JCS must contain a *CD statement so that DataRefresher knows where to place the generated control deck and column descriptors. The *CD statement follows a //SYSIN DD * statement in your JCS.

EXTDATA
DataRefresher places the load control deck at the beginning of the data set identified on the EXTDATA keyword. If CD=EXTDATA, you must also include the EXTDATA keyword.

Although this keyword has no effect on IMS DPROP, it is recommended that you provide it on your SUBMIT command, as it reduces the risk of errors; additionally, it is more efficient to have DataRefresher generate the load control deck or column descriptor than for you to provide it.

For more information on this keyword, refer to the appropriate DataRefresher or DXT documentation.

DECIMAL=PERIOD | COMMA (OPTIONAL)
Specifies the format of the decimal-point character in your extract output. Write DECIMAL=COMMA if you want the decimal point to be represented by a comma, DECIMAL=PERIOD if you want the decimal point to be represented by a period.

USERDECK=’ *optionlist* **’ (OPTIONAL)**

Explicitly provides DB2 load utility options. These options are included in the control deck generated by the DEM for DB2.

For the DB2 LOAD statement, DataRefresher generates the CONTINUEIF keyword, which indicates a value used for a continuation field in the input record. All other LOAD keywords use the DB2 default values unless otherwise specified by the DataRefresher USERDECK keyword.

optionlist

Represents selected DB2 parameters you want to include in the DataRefresher-generated LOAD commands. The DEM does not validate the *optionlist* variable. Errors will be detected by the DB2 load utility. For information on what options you can specify for the DB2 load utility, see the LOAD statement in DB2 Utility Guide and Reference.

The form of the USERDECK keyword is:

```
USERDECK='parm-1,...,parm-n'
```

where each of the *parm-n* specifications is replaced by a DB2 load utility option.

RESUME(NO) is the default for the USERDECK keyword.

Note: DataRefresher no longer automatically provides for RESUME(YES) on the DB2 LOAD statement; if you want your extract request output to be loaded into a table that already contains data, specify USERDECK=’RESUME(YES)’ in your corresponding requests.

ACCOUNT=’ *actinfo* **’ (OPTIONAL)**

Writes account information used by the exit routine if your installation uses an accounting exit routine to keep track of DEM use. The value that you write can be anything provided that it is no more than 32 characters and is enclosed in single quotation marks. Contact your DataRefresher administrator for specific information about a local standard accounting scheme.

NULLS=Y | N (OPTIONAL)

Is typically omitted or specified as NULLS=Y for the definition of IMS DPROP propagation requests.

For more information on this keyword, refer to the appropriate DataRefresher or DXT documentation.

TYPE=NORMAL | PERSIST (OPTIONAL)

Indicates the status of the extract request after it is processed by the DEM. This operand is valid only if you are using DXT Version 2 Release 5 or DataRefresher.

PERSIST

Indicates that the extract request remains in EXTLIB after successful execution by the DEM.

NORMAL

Indicates that the extract request is deleted from the EXTLIB (as it was in prior DataRefresher releases) after it is successfully processed by the DEM. This is the default value.

EXTRACT

The DataRefresher EXTRACT statement of the DataRefresher SUBMIT command identifies the fields to be extracted by DataRefresher and propagated by IMS

DPROP. The DataRefresher EXTRACT statement also identifies to or from which DB2 column of the propagated table each field is mapped.

This topic describes the DataRefresher EXTRACT statement as it is used with IMS DPROP. Refer to the appropriate DataRefresher or DXT documentation for complete information on the EXTRACT statement.

The syntax of the EXTRACT statement is shown in Figure 30.

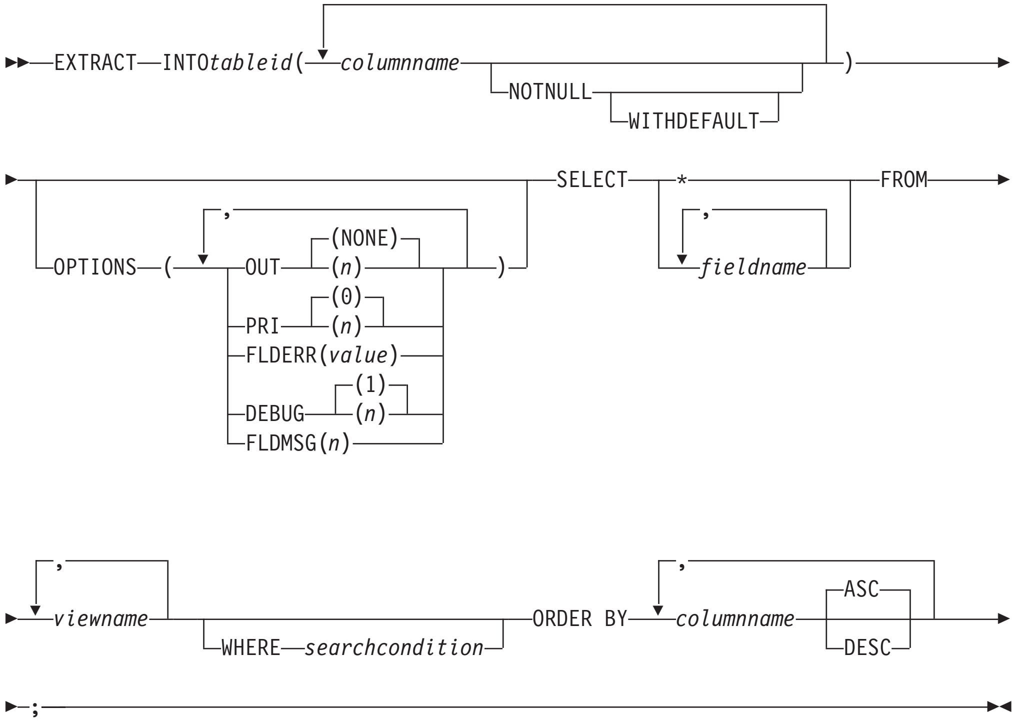

Figure 30. The EXTRACT Statement

EXTRACT (REQUIRED)
Extracts data from an IMS data source.

INTO *tableid (columnname)* **(REQUIRED for IMS DPROP)**
Identifies the name of the table and the table columns into which the extracted data is to be inserted and to or from which the changed data is propagated.

tableid
Specifies the ID of the table into which you want your data inserted. The *tableid* can be qualified or unqualified.

IMS DPROP checks to make sure that the table has been previously created and has a primary key.

tableid has the form *authid.tablename*.

authid
Is the authorization ID. If specified, it must be a valid DB2 authorization ID, as described in *DB2 SQL Reference*. If you:

- Provide an *authid*, the propagation request is referred to as a propagation request with a qualified table name.

  PR definitions with qualified table names are used typically for production environments. They support propagation to or from only *one* table: the table whose qualified table name is specified on the INTO clause of the EXTRACT statement.
- Do not provide an *authid*, the propagation request is referred to as a propagation request with an unqualified table name.

  If you do not use *authid* for DB2, the load utility assumes that it is the same as the user ID written in the JOB statement of the JCS data set associated with your ER. If no user ID was specified on the JOB statement, DB2 will use the default user ID that was defined when DB2 was installed for batch jobs.

  PR definitions without qualified table names are used in some test environments. They can support propagation to or from one of multiple, identically structured tables having the unqualified table name specified on the INTO clause of the EXTRACT statement.

For more information on propagation requests defined with and without table name qualifiers, refer to the appropriate Administrators Guide for your propagation mode.

tablename
Specifies the table into which the extracted data is loaded and to or from which the changed data is propagated. The *tablename* must be written. It can be either an SQL name, an SQL-quoted name, or a DBCS name.

If you do not write an authorization ID, do not write the period in front of *tablename*.

For a generalized mapping case propagation request, the table name must be unique across all propagation requests on the mapping tables of the IMS DPROP directory; that is, the table can only be propagated by one propagation request. A more detailed description of this rule follows:

- If the same table name is specified once on one propagation request with a qualifier and another time on another propagation request without a qualifier, IMS DPROP considers that both propagation requests propagate the same table. IMS DPROP therefore considers this an error.
- If the same table name is specified without qualifiers on two different propagation requests, IMS DPROP considers that both propagation requests propagate the same table. IMS DPROP also considers this an error.
- If the same table name is specified on two different propagation requests with two different qualifiers identifying two different propagated tables, no error exists.

For user mapping cases, several propagation requests can propagate data to or from the same table. However, tables cannot be propagated by a generalized mapping case propagation request and a user mapping case propagation request.

For DB2-to-IMS and two-way synchronous propagation, you cannot specify the same table name in one propagation request as qualified and in another as unqualified. You must specify a specific table name either always *with* a qualifier, or always *without* a qualifier.

columnname

Specify the names of the columns into which the extracted data is to be inserted and to or from which the changed data is to be propagated.

For a propagation request belonging to a generalized mapping case, all columns of the primary DB2 key should be included in this list of column names, as well as all columns defined to DB2 as NOT NULL.

Columns mapped from fields located in extension segments of a mapping case 2 propagation request should be defined to DB2 as NOT NULL WITH DEFAULT or nullable. Columns mapped from fields located in variable-length segments that could be missing in some segment occurrences should also be defined to DB2 as NOT NULL WITH DEFAULT or nullable.

The DB2 limit for *columnnames* depends on the release level, but is at least 300. Refer to the DB2 manuals for information on the current limit at your installation. A *columnname* can be either an SQL name, an SQL-quoted name, or a DBCS name. If you write more than one column name, separate them with commas.

Specifying nulls: Specifying NOT NULL or NOT NULL WITH DEFAULT for the columns in the INTO clause is used by DataRefresher but not by IMS DPROP. IMS DPROP gets this information from the DB2 catalog.

To achieve the same mapping during the DataRefresher extract and IMS DPROP propagation, describe the null attribute of each DB2 column the same way to DataRefresher (on the INTO clause of the EXTRACT statement) and to DB2 (on the CREATE TABLE statement).

If one or more columns cannot contain null values, you must follow the column name with NOT NULL. If you do not specify NOT NULL, the DEM assumes that the column can contain null data and produces a NULLIF statement in the generated control statement for the column.

If a column is defined as NOT NULL WITH DEFAULT, the DEM includes a DEFAULTIF keyword as part of the description of the relevant column when generating the DB2 LOAD control deck. This results in loading zeros into numeric fields and blanks into character/graphic fields wherever the source field is considered to be nonexistent; for example, fields located in extension segments of a mapping case 2 propagation request where the extension segments do not exist.

Ordering columns: The order of your column names is important and depends on how you write the SELECT statement and INTO statement.

The order you write the field names on the SELECT statement is the order they are inserted in the columns named on the INTO statement.

The number of columns you name on the INTO statement must be equal to the number of fields you name on the SELECT statement. You can select duplicate fields with the SELECT statement, but you cannot specify duplicate columns on the INTO statement.

Ordering columns when specifying SELECT *:

Specifying * for the SELECT statement will select all the fields from a source DXTVIEW. In this case, your target columns are matched with the fields contained in the relevant DXTVIEWs.

The number of columns you name on the INTO statement must be equal to the number of fields listed in the relevant DXTVIEWs.

Reasons for termination or cancellation:

If the number of columns specified on the INTO statement does not equal the number of fields of extracted data, the request is not queued for execution.

If the source data has null values and you are entering data into a table with the NOT NULL option, the DEM issues an error message and terminates.

OPTIONS (OPTIONAL)

Affect:

- How much output your extract request produces (OUT)
- When the DEM executes your extract request (PRI)
- How the DEM handles field errors when executing your extract request (FLDERR)
- How much diagnostic information the DEM maintains when executing your extract request (DEBUG,FLDMSG)

OUT(NONE | *n*) (OPTIONAL)

Controls the number of output rows the DEM can generate when executing your extract request:

NONE

Indicates that there is no limit on the number of output rows that the DEM can generate when executing your extract request. If you specify OUT(NONE), your extract request can only execute when the DEM also has an output limit of NONE, specified on the INITDEM command.

n Indicates the maximum number of output rows you want generated when the DEM executes your extract request.

If this number is reached, the DEM will stop writing your output, even if your output is not yet complete.

Replace *n* with a number from 1 through 10000000 (ten million), written without commas or spaces.

The OUT keyword allows your installation to schedule execution of your extract requests on a basis of size. For example, your installation could set up the DEM to run small extract requests during the day and large extract requests overnight.

PRI(0 | *n*) (OPTIONAL)

Controls when your extract request will be executed. Replace *n* with a number from 0 through 255.

Generally, the larger the priority number, the more quickly an extract request is executed. The DEM can execute an extract request only if the extract request's priority number falls within its priority range. See the appropriate DataRefresher or DXT documentation for more information about DEM priority ranges.

FLDERR(*value*)

Controls what the DEM does if it encounters a field with an invalid format while running your extract request. The DEM can encounter an incorrectly formatted field in a field named on the SELECT statement or a field named on the WHERE statement.

If a field named on the SELECT statement is incorrectly formatted, the DEM responds based on the information you specify on the FLDERR keyword.

The values you can specify with the FLDERR keyword are:

SUBST(NULL)
Tells the DEM to substitute nulls when it writes the contents of the fields causing the field errors.

SUBST(NULL), *n*
Tells the DEM to substitute nulls when it writes the contents of the fields causing the field errors and to stop executing your extract request after a given number of substitutions.

Replace *n* with the number (1 through 10000) of field errors that you will allow the DEM to substitute in your extract request. If the DEM encounters *n*+1 field errors, it will stop executing your extract request.

SUBST(ZERO)
Tells the DEM to substitute zeros when it writes the contents of numeric fields causing the field errors. The DEM will write blanks for any other type of field, such as character or graphic.

SUBST(ZERO), *n*
Tells the DEM to substitute zeros when it writes the contents of numeric fields causing the field errors and to stop executing your extract request after a given number of substitutions. The DEM will write blanks for any other type of field, such as character or graphic.

Replace *n* with the number (1 through 10000) of field errors that you will allow the DEM to substitute in your extract request. If the DEM encounters *n*+1 field errors, it will stop executing your extract request.

HALT
Tells the DEM to stop running your extract request when it encounters a field error.

SKIP
Tells the DEM to skip the record containing the field error and continue running your extract request.

SKIP, *n*
Tells the DEM to stop after skipping a given number of field errors while executing your extract request.

Replace *n* with the number (1 through 10000) of field errors that you want to allow the DEM to skip. If the DEM encounters *n*+1 field errors, it will stop executing your extract request.

Notes:

1. To avoid propagation failures, the IMS data copy and the DB2 data copy should be consistent when the extract is completed. Therefore, make sure that the DEM does not encounter field errors during the extract. Check after the extract is completed whether field errors have been encountered.
2. If the execution of an extract request is stopped because of a field error, the DEM writes all of the output resulting from the extract request up to the point of the field error and removes the extract request from the EXTLIB.
3. If you indicate that a target column cannot contain a null value, (with NOT NULL on the INTO statement) and if the selected field is packed decimal, zoned decimal, date/time, or user-defined, you must specify a FLDERR value other than (SUBST(NULL)) or (SUBST(NULL),*n*). (These types of data are the only types for which conversion errors are

possible.) If you specify that the DEM substitute nulls in that situation, the DEM will generate nulls, causing the load utility to terminate with an error.

4. If you specify FLDERR(SUBST(NULL)) or FLDERR(SUBST(NULL),*n*), and the column allows nulls, the DEM will put a NULL IF statement for each target column into the load command that it generates. If, however, one of the columns in the target table cannot contain a null value, the NULL IF statement will cause the load utility to terminate with an error. Therefore, if a target column cannot contain a null value, you must indicate a FLDERR value other than (SUBST(NULL)) or (SUBST(NULL),*n*).

When your job stops because of field errors, you can resubmit the job after:

- Fixing the error (if you specified FLDERR(HALT))
- Increasing the value of *n* (if you specified FLDERR(SKIP,*n*))
- Changing the keyword to FLDERR(SKIP) if you specified FLDERR(HALT) or FLDERR(SKIP,*n*)

DEBUG(*n*) (OPTIONAL)

Affects what diagnostic information the DEM maintains and issues as output when executing your extract request.

Replace *n* with a number from 1 through 4. See the appropriate DataRefresher or DXT documentation. for a description of the type of diagnostic information associated with each debugging level.

The default value for *n* is 1.

FLDMSG(*n*) (OPTIONAL)

Limits the number of field format errors to be diagnosed with messages. If more than *n* field format errors occur, no details will be presented on the *n+1*st and succeeding errors. Instead, a summary message telling how many field errors were encountered will be written.

FLDMSG(*n*) is written as a string of digits without commas or spaces, where *n* is any number from 0 through 1000000 (one million).

The default value for *n* is 100.

SELECT * | *fieldname* (REQUIRED)

Identifies the fields from which you want to extract data and from/to which you want to propagate the changed data.

*** (asterisk)**

Indicates that you want to extract data from all of the fields contained in the DXTVIEWs named in the EXTRACT statement's FROM keyword.

If you write * on the SELECT statement and the names of two DXTVIEWs with common field names on the FROM keyword, data will be extracted from all of the fields contained in the first DXTVIEW, but only from the nonoverlapping fields contained in the second DXTVIEW.

fieldname **save.fields**

Indicates the names of the fields from which you want to extract data. The fields must be contained in the DXTVIEWs named on the FROM keyword. If your name contains special characters, you must enclose it in double quotes (″).

Write *fieldname* as:

viewname.fieldname

where *fieldname* is the name (or alias) of the field from which you want to extract data and *viewname* is the name of the DXTVIEW containing the field. Use *viewname* if:

- You want to extract data from fields contained in two different DXTVIEWs, and
- The name (or alias) of the field you want to extract data from is identical to the name (or alias) of one of the fields contained in the other DXTVIEW.

If you do not write the view name, do not write the period in front of the field name.

You can write up to 300 *fieldnames* separated with commas. This is a DB2 limit. Depending on the release level at your installation, the limit can be higher than 300. Refer to DB2 Command Reference for information on the current limit at your installation.

Table 5 lists some rules to consider for selecting fields.

Table 5. Rules for Selecting Fields

For a propagation request belonging to mapping case...	Select fields....
1	• Located in the entity segment • Located in the IMS key of the parent/ancestors of the entity segment • Located in the data portion of the physical parent/ancestors of the entity segment if path data is selected
2	• Located in the extension segments • Located in the entity segment • Located in the IMS key of the parent/ancestors of the entity segment • Located in the data portion of the physical parent/ancestors of the entity segment if path data is selected
3	• Located in the entity segment which is an internal segment • Located in the IMS key of the containing parent segment (the segment that contains the internal segment) • Located in the IMS key of the parent/ancestors of the containing parent segment • Located in the data portion of the containing parent segment and of the physical parent/ancestors of the containing parent segment if path data is selected
user	• That you want to have extracted by DataRefresher during the IMS extract

IMS DPROP has some rules for the mapping of *key* fields. Typically, you map all fields that are within thefully concatenated key of an entity segment to the columns of a DB2 primary key. Refer to the appropriate *Administrators Guide* for your propagation mode for information on mapping rules for keys.

FROM *viewname* **(REQUIRED)**

Identifies the DXTVIEWs containing the fields from which you want to extract data.

Replace *viewname* with the names of the relevant DXTVIEWs. You can specify up to 16 views.

Separate multiple view names with commas. If your DXTVIEW name contains special characters, you must enclose it in double quotes (″).

Specifying multiple views: You can specify views based on different DataRefresher PCB descriptions using the same FROM keyword.

When you specify two or more views in a FROM keyword, DataRefresher extracts data from each view and joins it to form one output row.

The views you specify with the FROM keyword can be a combination of views to which you have access authorization. When you specify more than one view with one FROM keyword, you cannot specify views of PCBs in different PSBs. You can, however, specify views of different PCBs in the same PSB.

When you code the FROM keyword, IMS DPROP has the following requirements:

For a propagation request belonging to mapping case 1 or 3:
Specify only one DXTVIEW. For information on creating a DXTVIEW, refer to “CREATE DXTVIEW ” on page 45.

For a propagation request belonging to mapping case 2:
Specify one DXTVIEW for each extension segment type. All DXTVIEWs specified in the FROM clause should refer to the same DataRefresher PCB and DataRefresher PSB.

For a propagation request belonging to a user mapping case:
Each segment to be propagated by the propagation request must be in the hierarchical path of a DXTVIEW identified in the FROM clause. Depending on the type of DataRefresher mapping that you need for the IMS extract, DataRefresher can have additional requirements. For more information on these requirements, refer to the appropriate DataRefresher or DXT documentation.

ORDER BY (OPTIONAL)

Allows you to control the sequence of the EXTRACT output.

This function basically provides the same function as the ORDER BY clause of an SQL SELECT statement.

You can specify a maximum of 16 columns to determine the final sequence of the extract. You must SELECT each specified column.

WHERE *search condition* **(OPTIONAL)**

Specifies the conditions that the IMS data must satisfy to be extracted by the DataRefresher DEM for any specified mapping case and to be propagated by IMS DPROP when the propagation request is a generalized mapping case.

The IMS data is propagated to a DB2 table depending on IMS field values. There is no similar WHERE clause to propagate DB2 data depending on DB2 column values.

The WHERE clause is described more fully in “WHERE clause ” on page 70.

WHERE clause

Use the WHERE clause to specify under which conditions an occurrence of the segment type should be extracted by DataRefresher and propagated from IMS to DB2. For example, you can set a condition based on field values or a combination of field values.

With the generalized mapping logic of IMS DPROP and DataRefresher, you can optionally include a WHERE clause in the propagation request definition.

IMS DPROP supports WHERE clauses for propagation requests belonging to mapping case 1 and 2. WHERE clauses are not supported for mapping case 3 propagation requests. For more information on using the WHERE clause for DB2-to-IMS synchronous propagation, refer to the appropriate Administrators Guide for your propagation mode.

Syntax of the WHERE Clause

The syntax of the WHERE clause is shown in Figure 31. *operand1* and *operand2* can be field names or constants.

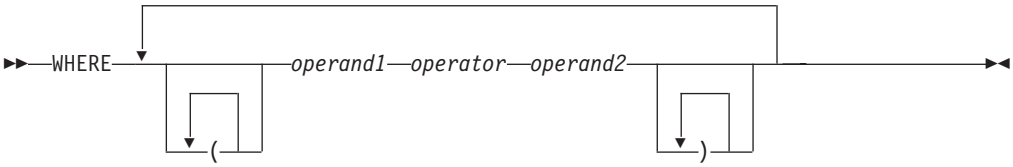

Figure 31. The WHERE Clause

Examples of the WHERE Clause

Below are some examples of how the WHERE clause can be written.

Note: The operators *AND* and *OR* are not mutually exclusive.

```
WHERE FIELD1 > FIELD2
WHERE FIELD1 => 'ABC'
WHERE FIELD1 => 'ABC' AND FIELD2 = 'CDE'
WHERE ((FIELD1 => 'ABC') AND (FIELD2 = 'CDE')) OR (FIELD4 = 4)
```

Figure 32. Examples of WHERE clauses

Naming the Fields

For TYPE=E propagation requests, all fields in the WHERE clause must be both:

- Listed within the SELECT statement
- Contained in the DXTVIEWs that you name in the FROM keyword of the EXTRACT statement

For other types of propagation requests, all fields in the WHERE clause must be contained in the DXTVIEWs that you name in the FROM keyword, but they do not have to be listed within the SELECT statement.

For DataRefresher releases prior to DataRefresher Version 2 Release 5, IMS DPROP has the following restrictions:

- Field names specified in the WHERE clause must be unique across all DXTVIEWs referenced in the FROM keyword of the EXTRACT statement. Do not qualify the field name with the name of the DXTVIEW.
- You cannot specify fields that have been assigned alias names in the DXTVIEWs in the WHERE clause.

For DXT Version 2 Release 5 and DataRefresher:

- IMS DPROP supports field names that are not unique across the DXTVIEWs in the WHERE clause.

  When your FROM keyword lists two DXTVIEWs that contain fields with the same name, and you want to include one of those fields on a WHERE clause, qualify the field with the name of its corresponding DXTVIEW.
- If you want to include a field that has been assigned an alias, use the alias, not the field name.

For more information on which fields can be included in the WHERE clause, see the appropriate Administrators Guide for your propagation mode.

Forming the Conditions

A single condition has the following format:

- *operand1 operator operand2*, where:

operand1 **and** *operand2*

Can be field names or constants, but at least one of the two operands must be a field name. The two operands must have compatible data types; both of them must have either a numeric, character, graphic, date, time or timestamp data type. The two operands must also have compatible lengths; that is, the field must be able to contain the value to which it is being compared.

operator

Performs a comparison between the two operands.

If the operands represent numeric data, arithmetic comparisons are made. If the operands represent character data, logical comparisons are made. DBCS fields are compared in EBCDIC collating sequence.

The comparisons that can be made with WHERE are:

= equal to

> greater than

>= greater than or equal to

< less than

<= less than or equal to

¬= not equal to

Unlike DataRefresher, IMS DPROP does not support LIKE, NOT LIKE, IN, NOT IN or BETWEEN.

Specifying Multiple Conditions

If you write more than one condition, separate them with the Boolean operators AND or OR. Unlike DataRefresher, IMS DPROP does not support the NOT Boolean operator.

The order of evaluation for Boolean operators is:

Order Operator

1 ()

2 AND

3 OR

Use parentheses to control the order in which Boolean expressions are evaluated.

Restrictions for Numerical Operands

IMS DPROP has the following restrictions for numerical operands in the WHERE clause:

- Unlike DataRefresher, IMS DPROP does not support floating point operands.
- For TYPE=E propagation requests, IMS DPROP does not allow you to use a numerical IMS field if the scale of the IMS field and the scale of its target DB2 column are different.
- If the two operands of a single condition have different scales, then the sum of the following should not exceed 31 digits:
  - The larger of the two scales, and
  - The larger of the two numbers of integer digits

  This sum should not exceed 31 digits because IMS DPROP internally converts the two operands to an intermediate field. This intermediate field has 31 digits, and its scale is the larger of the two scales of the two operands. The values of the two operands must fit in the intermediate field.

Rules for Special Cases

IMS DPROP has some rules for evaluating a single condition of the WHERE clause in the following cases:

Missing Fields and NULL Values: IMS DPROP evaluates the single condition as FALSE[2], if an operand used in a single condition of the WHERE clause is a field that either:

- Is missing in a segment occurrence (for example, because the field is located beyond the end of the variable length segment)
- Has a NULL value, because it is processed by a Field exit routine that has requested mapping to a DB2 NULL

Improper Data Values: If a field used in a single condition of the WHERE clause has a value that does not correspond to the data format defined to IMS DPROP (for example, a non-numerical value in a field defined as numerical), IMS DPROP typically considers the single condition FALSE. However, if the operator for the single condition is ÿ=, IMS DPROP evaluates the condition as TRUE.[3]

When evaluating a WHERE clause, the RUP does not consider a field containing an improper data value as an error and does not call its propagation error logic. This is because improper data values can be normal when an IMS segment contains redefined data.

2. While this logic does not necessarily match DataRefresher logic, it does match DB2 logic and SQL standards. During the processing of a row, if DB2 compares one operand in a WHERE clause with another operand having a NULL value, it evaluates the condition as UNKNOWN. For such a condition, DB2 considers the row not qualified.

3. While this logic does not necessarily match DataRefresher logic in all circumstances, it is consistent with IMS DPROP logic and DB2 logic used in UNKNOWN conditions. See previous footnote for further information.

WHERE Clause Constants

You can include numeric constants, character constants, or both in the WHERE clause. Character constants must be enclosed in single quotes. Date, time, and timestamp constants must also be enclosed in single quotes.

The value of a numeric or character constant that you write with WHERE must fall within the range of allowable values for its associated field type. When you write a constant that falls outside the allowable range for the field, the DataRefresher UIM does not allow the extract request to be successfully created. Table 6 lists the different field types and the allowable ranges for their constants.

Table 6. Field Types and Allowable Ranges

Field type	Data type	Sample field format	Allowable range for constant
A	date data	1990-9-1	8 to 10 bytes
B	single-byte binary data	12	1 through 255
C	EBCDIC character string	ABC@123	1 to 254 characters
F	32-bit, signed binary integer	(+/-)123	-2147483648 to +2147483647
G	DBCS data		1 to 127 DBCS characters; (2 to 254 bytes, must be even number)
H	16-bit, signed binary integer	(+/-)123	-32768 to +32767
P	packed decimal number	(+/-)123.456	31 significant digits
S	time stamp data	1991-2-5-10.45.23.01	16 to 26 bytes
T	time data	4.01	4 to 8 bytes
VC	variable-length character data	ABC@123	1 to 254 characters
VG	variable-length graphic data		1 to 127 DBCS characters; (2 to 254 bytes, must be even number)
Z	zoned decimal number	(+/-)123.456	16 significant digits

Unlike DataRefresher, IMS DPROP does not support floating point fields and floating point constants.

Date Constants: These must be enclosed in single quotes. IMS DPROP supports the following formats for date constants in the WHERE clause:

- ISO
- USA
- EUR
- JIS

Date constants must therefore be in one of the following formats:

yyyy-dd-mm ISO and JIS

dd.mm.yyyy EUR

mm/dd/yyyy USA

You can omit leading zeros from the month and day.

Time Constants: These must be enclosed in single quotes. IMS DPROP supports the following formats for time constants in the WHERE clause:
- ISO
- USA
- EUR
- JIS

Time constants must therefore be in one of the following formats:

hh.mm.ss ISO and EUR

hh:mm:ss JIS

hh.mm ***x*M** USA

Where *x* is either A or P.

You can omit:
- Leading zeros from the hour.
- Seconds entirely. In this case, an implicit specification of 0 seconds is assumed.

Timestamp Constants: These must be enclosed in single quotes and must be in the following format:

yyyy-mm-dd-hh.mm.ss.nnnnnn

You can:
- Omit leading zeros from the month, day, and hour
- Omit or truncate microseconds

Graphic Literals: If the WHERE condition contains a graphic literal (DBCS), the literal must be enclosed in single quotes and be preceded by a G. In addition, the data must be bracketed by a shift-out (X'0E') character and a shift-in (X'0F') character as in:

```
WHERE NAME=G'<D B C S N A M E>'
```

X'0E' is represented by < and X'0F' by >.

Graphic literals can contain only DBCS data; single-character data is not allowed in a graphic literal. In addition, graphic literals can only be compared with graphic fields. In the above example, NAME must be a graphic field.

Mixed Literals: Literals and fields containing both DBCS and single-byte data are not supported by IMS DPROP.

G Fields of Unequal Length: If a comparison is made between two G fields of unequal length, IMS DPROP pads the shorter DBCS fields with DBCS blanks (X'4040') when comparing the fields.

User Data Type: If a comparison is made with a user data type, the comparison is made after the user data type exit has converted the field from its user data type format into its target format in a DataRefresher data type. For example:

```
WHERE UDTFIELD='CHARACTER_OUTPUT'
```

In this WHERE condition, the field called UDTFIELD is defined with user data type format ZZ. The ZZ data type was defined as having a target data type of character (C) format of 16 bytes. Before the WHERE condition is evaluated, the user data

type exit for data type ZZ is called to convert this field into its target (character) data type. If there are no errors from the data type exit, the field is converted, and the data returned from the data type exit is compared with the WHERE condition 'CHARACTER_OUTPUT'.

JCL for Defining Propagation Requests with DataRefresher UIM

This topic describes the DataRefresher UIM JCL needed to process SUBMIT commands with a MAPEXIT=EKYMCE00 keyword.

When you provide a MAPEXIT=EKYMCE00 keyword on a SUBMIT command, DataRefresher UIM calls the IMS DPROP-provided DataRefresher Map Capture exit, EKYMCE00. EKYMCE00 runs as a DB2 application and needs to access the IMS DPROP directory tables and the DB2 catalog tables. You must provide DB2-related and IMS DPROP-related JCL statements in the JCL that calls DataRefresher UIM. You must provide JCL statements to precompile, assemble, and link-edit the SQL update modules that IMS DPROP generates for each propagation request belonging to a generalized mapping case.

The JCL for these sample procedures can be found in the EKYINST library.

DataRefresher UIM can call EKYMCE00 several times in one job step; it processes one propagation request during each call.

Environment

If you have been using:

- DXT 2.5 on its own and now want to use it with IMS DataPropagator Version 3 Release 1, use the JCL shown in Figure 33 on page 77 instead of the JCL distributed with DXT.
- DataRefresher GUI on its own and now want to use it with IMS DataPropagator Version 3 Release 1, the JCL of the job generated will run using the Call Attach facility (CAF Attach), which is controlled by the DataRefresher host component.
- DataRefresher on its own and submitting the propagation requests yourself (without using the GUI) and now want to use it with IMS DataPropagator Version 3 Release 1, use the JCL shown in Figure 33 on page 77, which uses TSO Attach to connect to DB2. The DataRefresher UIM must be called by a RUN CP subcommand of the DB2 DSN command.

EKYMCE00 executes in 31-bit addressing mode.

Examples of the JCL for DataRefresher UIM

In this topic, sample JCL procedures are provided for:

EKYUUIMP Procedure that the DataRefresher UIM uses to process SUBMIT commands that have a MAPEXIT=EKYMCE00 keyword. It contains control statements to execute DataRefresher UIM as a DB2 application under the TSO Attach facility and default propagation parameter values.

EKYUUIMJ JCL to execute EKYUUIMP.

EKYUDB2P Procedure to invoke the DB2 load utility.

EKYUDB2J JCS to execute EKYUDB2P.

EKYUUIMP - Sample JCL Procedure to Execute the DataRefresher UIM

Figure 33 on page 77 shows EKYUUIMP, which DataRefresher uses to process SUBMIT commands that have a MAPEXIT=EKYMCE00 keyword.

Before invoking the procedure, you must:

- Replace all occurrences of `1111111` with the high level qualifier(s) for the IMS DPROP libraries
- Replace all occurrences of `2222222` with the name of the DataRefresher high level qualifier(s)
- Replace all occurrences of `3333333` with the data set name of the IMS DBDLIB
- Replace all occurrences of `4444444` with the data set name of the library that contains the DB2 DBRMs of the SQL update modules associated with the propagation requests
- Replace all occurrences of `5555555` with the data set name of the library that contains the SQL update modules associated with the propagation requests
- Replace all occurrences of `6666666` with the data set name of the DB2 load job destination
- Replace all occurrences of `7777777` with the data set name of the DB2 MACLIB
- Replace all occurrences of `8888888` with the data set name of the IMS MACLIB
- Replace all occurrences of `9999999` with the data set name of the library that contains DPROPUIN
- Replace all occurrences of `AAAAAAA` with the DataRefresher UIM input control data set

```
//*-------------------------------------------------------------------*
//*            DPROP VERSION 2 RELEASE 1
//*   PROCEDURE EKYUUIMP TO EXECUTE THE DXT-UIM FOR DPROP            *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  IF YOU ARE NOT USING THE DPROP INSTALLATION PANELS,CHANGE THE    *
//*  FOLLOWING VALUES TO COMPLY WITH YOUR INSTALLATION STANDARDS:-
//*  ------------------------------------------------------------   *
//*                                                                   *
//*  1111111   = DPROP HLQ(S)                                         *
//*  2222222   = DXT HLQ(S)                                           *
//*  3333333   = DBD LIBRARY                                          *
//*  4444444   = DBRM LIBRARY FOR SQL UPDATE MODULE                   *
//*  5555555   = LOAD LIBRARY FOR SQL UPDATE MODULE                   *
//*  6666666   = DPROP MACLIB                                         *
//*  7777777   = DB2 MACLIB                                           *
//*  8888888   = IMS MACLIB                                           *
//*  9999999   = LIB CONTAINING DPROPUIN                              *
//*  AAAAAAA   = DXT UIM INPUT CONTROL STATEMENT DATASET              *
//*                                                                   *
//*-------------------------------------------------------------------*
//*
//EKYUUIMP PROC EKYPREF=1111111,
//              DXTPREF=2222222,
//              DBDLIB=3333333,
//              DBRMLIB=4444444,
//              LOADLIB=5555555,
//              MACLIB=6666666,
//              DB2MAC=7777777,
//              IMSMAC=8888888,
//              JCLLIB=9999999,
//              DXTCNTL=AAAAAAA
//*-------------------------------------------------------------------*
//UIM          EXEC     PGM=IKJEFT01
//EKYRESLB     DD       DISP=SHR,
//             DSN=&EKYPREF..EKYRESLB
```

Figure 33. EKYUUIMP: Sample JCL Procedure for DataRefresher UIM (Part 1 of 2)

```
//*                                                                    *
//*-------------------------------------------------------------------*
//*  ASYNC CONSIDERATION:                                              *
//*  --------------------                                              *
//*  IN ASYNC MODE, THERE IS NO STATUS FILE FOR THIS DPROP SYSTEM      *
//*  AND THE "EKYSTATF" DD STATEMENT CAN THEREFORE BE REMOVED.         *
//*                                                                    *
//EKYSTATF     DD       DISP=SHR,
//             DSN=&EKYPREF..STATFILE
//*-------------------------------------------------------------------*
//FDTLIB       DD       DISP=SHR,
//             DSN=&DXTPREF..FDTLIB
//EXTLIB       DD       DISP=SHR,
//             DSN=&DXTPREF..EXTLIB
//DDJCS01      DD       DISP=SHR,
//             DSN=&DXTPREF..JCS
//DXTPRINT     DD       SYSOUT=*
//DXTOUT       DD       SYSOUT=*
//SYSIN        DD       DISP=SHR,
//             DSN=&DXTCNTL..UIMIN
//MVGPRINT     DD       SYSOUT=*
//SYSTSPRT     DD       SYSOUT=*
//SYSTSIN      DD       DISP=SHR,
//             DSN=&JCLLIB(DPROPUIN)
//DBDLIB       DD       DISP=SHR,
//             DSN=&DBDLIB
//MVGUMOD      DD       DSN=&&MVGUMOD,UNIT=SYSDA,
// DCB=(LRECL=80,BLKSIZE=6160,RECFM=FB),SPACE=(CYL,3)
//PSYSPRT      DD       SYSOUT=*
//DBRMLIB      DD       DISP=SHR,
//             DSN=&DBRMLIB
//SYSTERM      DD       SYSOUT=*
//ASMIN        DD       DSN=&&ASMIN,UNIT=SYSDA,
// DCB=(LRECL=80,BLKSIZE=6160,RECFM=FB),SPACE=(CYL,3)
//ASYSPRT      DD       DUMMY,DCB=BLKSIZE=80
//ASYSLIB      DD       DSN=SYS1.MACLIB,DISP=SHR
//             DD       DISP=SHR,
//             DSN=&MACLIB
//             DD       DISP=SHR,
//             DSN=&DB2MAC
//             DD       DISP=SHR,
//             DSN=&IMSMAC
//ASYSLIN      DD       DSN=&&OBJECT,UNIT=SYSDA,
// DCB=(BLKSIZE=720,RECFM=FBS,BUFNO=30),SPACE=(CYL,3)
//LSYSPRT      DD       SYSOUT=*
//SYSLMOD      DD       DISP=SHR,
//             DSN=&LOADLIB
//SYSUT1       DD       UNIT=SYSDA,SPACE=(CYL,3)
//SYSUT2       DD       UNIT=SYSDA,SPACE=(CYL,3)
//SYSUT3       DD       UNIT=SYSDA,SPACE=(CYL,3)
//*-------------------------------------------------------------------*
```

Figure 33. EKYUUIMP: Sample JCL Procedure for DataRefresher UIM (Part 2 of 2)

The //EKYRESLB and //EKYSTATF DD statements are common to many IMS DPROP components and are described in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The remaining statements are described below.

The PROC Statement

The PROC statement defines the various keyword parameters of the JCL procedure. When invoking this JCL procedure, you must provide values for the keywords listed below. The data set names for these keywords must be enclosed in single quotes.

EKYPREF The high-level qualifiers for the IMS DPROP libraries

DXTPREF The high-level qualifiers for the DataRefresher libraries

The data set name of a file containing the JCS for the execution of the DB2 load utility.

DBDLIB The data set name of the IMS DBDLIB.

DBRMLIB The data set name of the library containing the DB2 DBRMs of the SQL update modules associated with the propagation requests.

LOADLIB The data set name of the library containing the SQL update modules associated with the propagation requests.

MACLIB The data set name of the IMS DPROP MACLIB.

DB2MAC The data set name of the DB2 MACLIB.

IMSMAC The data set name of the IMS MACLIB.

The EXEC Statement
Calls TSO batch to execute DataRefresher UIM under DB2 with the TSO Attach.

//FDTLIB DD Statement
Describes the DataRefresher FDTLIB data set.

//EXTLIB DD Statement
Describes the DataRefresher EXTLIB data set.

/DDJCS01
Describes the file that contains the JCS for the execution of the DB2 load utility. The DD name of //DDJCS01 is specified in the JCS keyword of the SUBMIT command.

//DXTPRINT DD Statement
Describes the print file that DataRefresher UIM uses to write messages.

//DXTOUT DD Statement
Describes a file that DataRefresher uses for printing purposes.

//SYSIN
Contains the SUBMIT parameters.

//MVGPRINT DD Statement
Contains the messages issued by the MVG, MVGU, and MCE.

//MVGPRINT is opened and closed each time a propagation request is processed. Therefore, when //MVGPRINT specifies a data set other than a sysout data set, it must be allocated with DISP=MOD.

//SYSTSPRT DD Statement
Describes the print file that TSO batch uses to write messages.

//SYSTSIN DD Statement
Describes the input file to TSO batch. //SYSTSIN contains a DSN command, a RUN CP subcommand, the DXT UIM program name that TSO processes as a TSO command, and an END command.

This data set is primarily used to execute the DXT UIM as a DB2 application under TSO batch.

If more than one DB2 subsystem is installed, IMS DPROP uses this data set to get the name of the correct DB2 subsystem to perform the BIND PACKAGE or FREE PACKAGE of the SQL update module.

Figure 34 on page 81 shows a sample member containing the TSO input statements used to execute the DXT UIM.

IMS DPROP can only read this data set if it specifies an MVS sequential file or a member of a partitioned data set (for example, the member DPROPUIN), not an instream data set.

If //SYSTSIN specifies an instream data set, IMS DPROP uses the name of the default DB2 subsystem specified at DB2 installation (generally DSN).

//DBDLIB DD Statement
Specifies the DBD library that MVG accesses to validate and complete the database and segment information specified in the propagation request to process.

//MVGUMOD DD Statement
Contains the source code of the SQL update modules that MVG creates. It is also used as input for the DB2 precompiler. //MVGUMOD replaces the usual //SYSIN statement of the precompiler.

//PSYSPRT DD Statement
Contains the print output of the DB2 precompiler of the SQL update module. //PSYSPRT replaces the usual //SYSPRINT statement of the DB2 precompiler.

//DBRMLIB DD Statement
Describes the DBRM library where the DB2 precompiler stores the DBRM of the generated SQL update module. Do not provide the member name on the JCL statement because MVG provides the name internally. The DBRM name provided by MVG is the same name as the propagation request ID. You will need to use this library when you bind the plans of the applications that perform propagation.

//SYSTERM DD Statement
Describes a print output file of the DB2 precompiler.

//ASMIN DD Statement
Contains the source code of the SQL update module after it has been processed by the DB2 precompiler. //ASMIN replaces the //SYSCIN statement of the precompiler and the //SYSIN statement of the assembler.

//ASYSPRT DD Statement
Contains the assembler listing of the SQL update module. //ASYSPRT replaces the usual //SYSPRINT statement of the assembler.

//ASYSLIB DD Statement
Describes the macro libraries required to assemble the SQL update module; that is, MACLIB for DB2, IMS DPROP, and MVS/ESA. //ASYSLIB replaces the //SYSLIB statement of the assembler.

//ASYSLIN DD Statement
Describes the object module of the SQL update module, which is passed to the linkage editor. //ASYSLIN replaces the //SYSLIN statement of the assembler.

//LSYSPRT DD Statement
Contains the print output of the linkage editor for the SQL update module. //LSYSPRT replaces the usual //SYSPRINT statement of the linkage editor.

//SYSLMOD DD Statement
Describes the load library where the linkage editor stores the load module of

the SQL update module. Do not specify the member name of the load module in your JCL, because MVG provides it internally. The load module name provided by MVG is the same name as the propagation request ID. The job steps performing propagation must be able to access the load module stored in this library.

//SYSUT1, //SYSUT2, and //SYSUT3 DD Statements
These three statements describe work files used by the DB2 precompiler, the assembler, and the linkage editor.

Sample DPROPUIN Member to Execute the DXT UIM

Figure 34 illustrates a sample DPROPUIN member containing control statements to execute the DXT UIM as a DB2 application under TSO Attach. This member is allocated by the //SYSTSIN DD statement in the DPROPUIM JCL procedure.

```
DSN SYSTEM( dsn)
 RUN CP PLAN(plan name)
  DVRU0000 EXITLANG=(LE)
 END
```

Figure 34. Sample DPROPUIN JCL to Execute DXT UIM

This sample should be adapted to your installation needs:

- Replace *dsn* with the name of your DB2 subsystem.
- The CP option is mandatory under the RUN subcommand and must be coded as shown.
- Replace *plan name* with the name of the DB2 plan used to execute DXT UIM with IMS DPROP.
- DVRU0000 is the program name of DXT UIM and must be coded as shown.
- Specify EXITLANG=(LE) only if DXT calls LE/370 high-level language exit routines during DXT UIM processing. (The DXT UIM can issue definition calls to LE/370 field exit routines during the processing of CREATE DXTPSB statements.)

If more than one DB2 subsystem is installed, the data set specified with the //SYSTSIN DD statement should be an MVS sequential file or a member of a partitioned data set (for example the member DPROPUIN). This is so IMS DPROP can get the name of the correct DB2 subsystem to perform the BIND PACKAGE or the FREE PACKAGE function.

EKYUUIMJ - Sample JCL to Call EKYUUIMP

Figure 35 on page 82 illustrates the JCL statements to call EKYUUIMP for the execution of the DataRefresher UIM.

Before submitting the JCL, you must:

- Replace `jjjjjjj` with valid jobcard details
- Replace all occurrences of `1111111` with the name of your DB2 subsystem

```
//JJJJJJJ JOB     *** VALID JOB CARD DETAILS ***
//*
//*-------------------------------------------------------------------*
//*                DPROP VERSION 2 RELEASE 1                          *
//*       JOB EKYUUIMJ TO EXECUTE THE EKYUUIMP PROCEDURE              *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*    ADAPT THE JCL AS FOLLOWS :                                    *
//*    --------------------------                                    *
//*                                                                  *
//*   JJJJJJJ  - JOB NAME                                            *
//*   1111111  - PACKAGE COLLECTION ID                               *
//*                                                                  *
//*   ADAPT THE MVGPARM CONTROL STATEMENT AS REQUIRED                *
//*******************************************************************
//RUNUIM   EXEC EKYUUIMP
//MVGPARM  DD  *
PRTYPE=L,MAPCASE=1,MAPDIR=HR,ERROPT=BACKOUT,
BIND=(PACKAGE(1111111) ISOLATION(CS) VALIDATE(BIND));
//*
```

Figure 35. EKYUUIMJ: Sample JCL to Call EKYUUIMP

The EXEC Statement

Calls TSO batch to execute the DataRefresher UIM with the TSO Attach.

The //MVGPARM DD Statement

Contains the default values for the propagation parameters in the MAPUPARM keyword of the DataRefresher SUBMIT command. You can replace the values in the sample with your own values. See Chapter 5, “//MVGPARM Data Set Propagation Parameters,” on page 133 for details of the default propagation parameters.

EKYUDB2P - Sample JCL Procedure to Execute the DB2 Load Utility

Figure 36 on page 83 shows the JCL to execute the DB2 load utility.

Before invoking the procedure, you must replace all occurrences of `1111111` with the name of your DB2 subsystem.

```
//*-------------------------------------------------------------------*
//*           DPROP VERSION 2 RELEASE 1                               *
//*  PROCEDURE EKYUDB2P TO EXECUTE THE DB2 LOADER                     *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  CHANGE AS REQUIRED:                                              *
//*  -------------------                                              *
//*                                                                   *
//*  1111111   - DB2 SUBSYSTEM NAME                                   *
//*                                                                   *
//*-------------------------------------------------------------------*
//EKYUDB2P     PROC      PRM=1111111
//LOADSTEP     EXEC      PGM=DSNUTILB,PARM='&PRM'
//UTPRINT      DD        SYSOUT=*
//SYSPRINT     DD        SYSOUT=*
//SORTWK01     DD        UNIT=SYSDA,SPACE=(CYL,2)
//SORTOUT      DD        UNIT=SYSDA,SPACE=(CYL,2)
//SORTWK02     DD        UNIT=SYSDA,SPACE=(CYL,2)
//SYSUT1       DD        UNIT=SYSDA,SPACE=(CYL,2)
```

Figure 36. EKYUUDB2P: Sample JCL Procedure to Execute the DB2 Load Utility

These statements are the standard JCS required to call the DB2 load utilities. Refer to DB2 Command Reference for a description of these statements.

EKYUDB2J - Sample JCL to Call EKYUDB2P

Figure 37 on page 84 shows the JCL to call the EKYUDB2P procedure.

Before submitting the JCL, you must replace `jjjjjjj` with valid jobcard details.

```
//JJJJJJJ  JOB   ***  VALID JOB CARD DETAILS  ***
//*-------------------------------------------------------------------*
//*               DPROP VERSION 2 RELEASE 1                           *
//*       JOB TO EXECUTE THE DB2 LOADER PROC (EKYUDB2P)               *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//* AMEND THE FOLLOWING VALUES TO SUIT YOUR INSTALLATION STANDARDS:-  *
//*                                                                   *
//*     JJJJJJJ  - JOB NAME
//*                                                                   *
//*-------------------------------------------------------------------*
//DB2LOAD      EXEC      EKYUDB2P
//SYSIN    DD   *
*CD
//SYSREC   DD   *
*EO
```

Figure 37. Sample DPROPARM JCL to Execute DataRefresher UIM

***CD** DataRefresher replaces the *CD statement with the DB2 load control statements.

The *CD record is required only if you specified CD=JCS on the DataRefresher SUBMIT command to request that DataRefresher write the control statements to an instream data set of this job (as opposed to a regular DASD data set).

***EO** DataRefresher replaces the *EO record with the extracted data.

The *EO record is required if you did not specify an EXTDATA keyword on the DataRefresher SUBMIT command. Refer to “SUBMIT/EXTRACT ” on page 49 for a brief description of the DataRefresher SUBMIT command or to the appropriate DataRefresher or DXT documentation for a detailed description.

DataRefresher DEM Commands

If you use the DataRefresher DEM to extract IMS data and load DB2 propagated data, you do not need to write your own extract programs. Using DataRefresher produces mapping and conversions that are identical to those performed by IMS DPROP during the propagation.

If you use DataRefresher in the extract and load phase of propagation:

- The mapping is based on information stored in the DataRefresher EXTLIB and FDTLIB.

  Use INITDEM, USE DXTPCB, and USE EXTID commands to control which DataRefresher ERs should be processed by the DEM.
- The DEM invokes EKYMCE00 once for each propagation request being extracted.

For DBRC-registered, full-function databases, EKYMCE00 verifies that the database has a read-only status and is not being concurrently updated by any IMS subsystem. This verification is only possible if DBRC share control is in effect.

- The DEM provides the extracted and mapped IMS data to the DB2 load utility.

  If you are using Segment exit routines, Field exit routines, or both, DEM calls them during the extract process.

  DEM does not call Propagation exit routines during the extract process; however, IMS DPROP calls them during data propagation. During an extract with the DEM, DataRefresher performs mapping and conversion, not your Propagation exit routines.
- After the extract is completed, DataRefresher deletes the processed ERs from the EXTLIB (unless the ER has been defined to DataRefresher with the TYPE=PERSIST keyword of the SUBMIT command).

You can ask the DataRefresher DEM to create the control statements for the DB2 load utility by providing a CD keyword on the DataRefresher SUBMIT command. You can also ask DataRefresher to create a job to execute the DB2 load utility by specifying the ddname of a file containing skeleton JCL for the DB2 load utility on the JCS keyword of the DataRefresher SUBMIT command.

If you are extracting or loading data into multiple tables, you might want to run the DB2 load utility jobs in parallel. This can reduce the amount of elapsed time required for loading the tables.

If the DB2 tables are involved in RIRs, you might want to specify the ENFORCE NO option on the USERDECK keyword of the DataRefresher SUBMIT command.

When you specify ENFORCE NO, the DB2 load utility does not check referential integrity constraints during the load processing. When the target table spaces have been loaded, DB2 places them in a “check pending state.”

After completing all DB2 load jobs, execute the DB2 check utility.

Executing the DataRefresher DEM: INITDEM, USE DXTPSB, and USE EXTID

To execute the DataRefresher DEM, provide the following commands in the DEM’s DXTIN data set:

- INITDEM
- USE DXTPSB or USE EXTID

An example of these commands is shown in Figure 38.

```
INITDEM NAME= demname, EXITLANG=(LE) ;
USE DXTPSB=psbname ;
USE EXTID=extid ;
```

Figure 38. The INITDEM and USE DXTPSB Commands

Replace *demname* with the name that you want to assign to the DEM. The name must consist of alphanumeric and special characters. To reduce confusion about the identities of a DEM, you should name a DEM based on the JCL job name.

Specify EXITLANG=(LE) only if DataRefresher calls LE/370 high-level language exit routines for the extract.

Replace *psbname* with the PSBNAME you assigned on the NAME keyword of the CREATE DXTPSB command. The USE DXTPSB command instructs DataRefresher to select ERs currently in the EXTLIB, whose definitions are based on the named DXTPSB.

Replace *extid* with extract ID you assigned on the EXTID= keyword of the SUBMIT command. The USE EXTID command instructs DataRefresher to perform the extract for a specific ER.

For more information on these commands, refer to the appropriate DataRefresher or DXT documentation.

JCL to Extract Data with DataRefresher DEM

This topic describes the DataRefresher DEM JCL needed to process ERs that have been defined with a MAPEXIT=EKYMCE00 keyword.

When processing an ER defined with the MAPEXIT=EKYMCE00 keyword, the DataRefresher calls EKYMCE00. You need to provide:

- IMS DPROP-related JCL statements in the JCL used to invoke the DataRefresher DEM
- JCL statements that IMS DPROP requires to call the IMS/ESA DBRC utility

The DataRefresher DEM can call EKYMCE00 several times in one job step; once for each processed ER with a MAPEXIT=EKYMCE00 keyword.

Environment

The DataRefresher DEM runs as an IMS batch or BMP application.

EKYMCE00 executes in 31-bit addressing mode.

Examples

Sample JCL procedures are provided for:

- EKYUDEMP, which is used to execute the DataRefresher DEM as an IMS batch program for ERs with a MAPEXIT=EKYMCE00 keyword.

  You can also execute the DataRefresher DEM as an IMS BMP.
- The JCL used to call the sample EKYUDEMP JCL procedure.

  As the JCL contains only an EXEC statement, it is not included on the sample JCL library.

Figure 39 on page 87 illustrates a sample JCL procedure for DataRefresher DEM.

Before invoking the procedure, you must:

- Replace all occurrences of `1111111` with the name of the IMS PSB used to execute the DEM as an IMS batch job
- Replace all occurrences of `2222222` with the data set name of the IMS PSBLIB
- Replace all occurrences of `3333333` with the data set name of the IMS DBDLIB
- Replace all occurrences of `4444444` with the data set name of the DataRefresher FDTLIB

- Replace all occurrences of `5555555` with the data set name of the DataRefresher EXTLIB
- Replace all occurrences of `6666666` with the data set name of the IMS DPROP MACLIB

```
//*-------------------------------------------------------------------*
//*             DPROP VERSION 1 RELEASE 1                             *
//*    PROCEDURE EKYUDEMP TO EXECUTE THE DXT-DEM WITH DPROP           *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  IF YOU ARE NOT USING THE DPROP INSTALLATION PANELS, YOU HAVE     *
//*  TO ADAPT FOLLOWING VALUES TO YOUR INSTALLATION'S STANDARDS:      *
//*  ---------------------------------------------------------       *
//*
//*   1111111   =   PSB NAME
//*   2222222   =   PSB LIBRARY
//*   3333333   =   DBD LIBRARY
//*   4444444   =   DPROP HLQ(S)
//*   5555555   =   DXT HLQ(S)
//*   6666666   =   DB2 LOAD JOB DESTINATION
//*-------------------------------------------------------------------*
//*
//EKYUDEM      PROC PSBNAME=1111111,
//             PSBLIB=2222222,
//             DBDLIB=3333333,
//             EKYPREF=4444444,
//             DXTPREF=5555555,
//             DXTOUT=6666666
//*-------------------------------------------------------------------*
//DEM          EXEC     PGM=DFSRRC00,
// PARM=(DLI,DVRX0000,&PSBNAME,20,0000,,0,,N,0,T)
//EKYRESLB     DD       DISP=SHR,
//             DSN=&EKYPREF..EKYRESLB
//IMS          DD       DISP=SHR,
//             DSN=&PSBLIB
//             DD       DISP=SHR,
//             DSN=&DBDLIB
```

Figure 39. EKYUDEMP: Sample JCL Procedure for DataRefresher DEM (Part 1 of 2)

```
//MVGPRINT     DD       SYSOUT=*
//DXTPRINT     DD       SYSOUT=*
//DXTOUT1      DD       &DXTOUT,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//FDTLIB       DD       DISP=SHR,
//             DSN=&DXTPREF..FDTLIB
//EXTLIB       DD       DISP=SHR,
//             DSN=&DXTPREF..EXTLIB
//SYSIN        DD       DSN=&&SYSIN,UNIT=SYSDA,
// DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),SPACE=(800,(10,2),RLSE)
//SYSPRINT     DD       DSN=&&SYSPRINT,UNIT=SYSDA,
// DCB=(LRECL=121,BLKSIZE=1210,RECFM=FBA),SPACE=(1210,(15,15),RLSE)
//*-------------------------------------------------------------------*
```

Figure 39. EKYUDEMP: Sample JCL Procedure for DataRefresher DEM (Part 2 of 2)

JCL Statements

The //EKYRESLB DD statements is common to many IMS DPROP components and is described in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The remaining DD statements are described below.

The PROC Statement

Defines the various keyword parameters of the JCL procedure. When invoking this JCL procedure, you must provide values for the keywords described below. Data set names identified on these keywords must be enclosed in single quotes.

PSBNAME Identifies the IMS PSB used to execute the DEM as an IMS batch job.

DataRefresher has the following PSBGEN requirements:

- The PSB must be generated for assembler or COBOL (the LANG keyword of the PSBGEN statement should be ASSEM or COBOL).
- The PSBGEN statement should specify CMPAT=YES.
- The order of the database PCBs within the IMS PSB should match the order of the DXTPCBs within the DXTPSB.
- The PCB statements for database PCBs should contain a POS=M keyword to request multiple positioning.
- If the PCBs do not match the DXTPSB, you must provide dummy PCBs.
- The PCB processing options should include processing options G and P (for non-Fast Path).

PSBLIB The data set name of the IMS PSBLIB.

DBDLIB The data set name of the IMS DBDLIB.

FDTLIB The data set name of the DataRefresher FDTLIB.

EXTLIB The data set name of the DataRefresher EXTLIB.

DXTOUT JCL specifications for the //DXTOUT1 DD statement. The DEM uses //DXTOUT1 to create JCL to call the DB2 load utility.

The EXEC Statement

Runs the DEM application (DVRX0000) under IMS batch.

//IMS DD Statement
Describes the IMS PSBLIB and IMS DBDLIB that execute the DEM as a standard IMS batch job.

//MVGPRINT DD Statement
Contains the messages issued by the MVG, MVGU, and MCE. This data set is opened and closed each time a propagation request is processed. Therefore, when MVGPRINT specifies a data set other than a SYSOUT data set, it must be allocated with DISP=MOD.

//DXTPRINT DD Statement
Describes a file that DataRefresher uses to print messages.

//FDTLIB DD Statement
Describes the DataRefresher FDTLIB data set.

//EXTLIB DD Statement
Describes the DataRefresher EXTLIB data set.

//SYSIN DD Statement
Contains generated input commands to the IMS DBRC utility, which is called by the MCE. These DBRC commands are generated by the MCE.

//SYSIN must be allocated to a temporary data set and *not* allocated as a SYSIN file.

//SYSPRINT DD Statement
Contains the print output of the IMS DBRC utility called by the MCE.

//SYSPRINT must be allocated to a temporary data set and *not* allocated as a SYSOUT file.

Chapter 4. Defining Propagation Requests Using MVG Input Tables

This topic describes the MVG input tables and how to define propagation requests with them.

The following topics provide additional information:
- “The Propagation Request MVG Input Table (DPRIPR) ”
- “The WHR MVG Input Table (DPRIWHR) ” on page 97
- “The SEG MVG Input Table (DPRISEG) ” on page 99
- “The TAB MVG Input Table (DPRITAB) ” on page 108
- “The FLD MVG Input Table (DPRIFLD) ” on page 109
- “Mapping Verification and Generation Utility (MVGU)” on page 115

The Propagation Request MVG Input Table (DPRIPR)

The propagation request table (DPRIPR) contains general information about the propagation request you are creating. For example, you would indicate in the propagation request table the type of propagation request you want to create, the mapping case you are using, and the direction in which you want to propagate your data.

The columns in the propagation request table are described in Table 7, with the defaults in bold and underlined.

Table 7. Columns of the propagation request MVG Input Table and their Data Types

Column	Data type	Description
PRID	char(8)	Propagation Request ID
USERID	char(8)	User ID
NODEID	char(8)	Node ID
PRTYPE	char(1)	PR type (E|L|F|U)
MAPCASE	char(1)	Mapping case (1|2|3)
MAPDIR	char(2)	Mapping direction (HR|TW|RH)
TABQUAL2	char(8)	Table qualifier used for validation
ERROPT	char(8)	Error option (BACKOUT|IGNORE)
MAXERROR	integer	Number of error messages
ACTION	char(4)	Add a new propagation request (**ADD**) or replace a propagation request (REPL)
PRSET	char(8)	Set of propagation requests
PROPSUP	char(1)	Selective suppression of propagation (Y|**N**)
EXITNAME	char(8)	Propagation exit name (if PRTYPE is U)
PROCSED	char(1)	Status of source propagation request in MVG input tables (Y = propagation request already processed, **not Y**: propagation request is not yet processed)
KEYORDER	char(3)	Ordering sequence of the key columns (**ASC**|DSC|ANY)
BIND	varchar(254)	Bind options for the package of the SQL update module

Table 7. Columns of the propagation request MVG Input Table and their Data Types (continued)

Column	Data type	Description
PATH	char(6)	Kind of path data selected in the PR, if any (ID|DENORM|**blank**)
AVU	char(1)	Avoid unnecessary updates (Y|N|**blank**)
DEFVEXT	char(1)	Default value extension segment (N|**Y or blank**)
COMMENT	varchar(254)	Comments about the propagation request for documentation purpose

PRID (REQUIRED)

The propagation request ID must be a valid load module name. It cannot be the same name as other load modules in your installation. IMS DPROP generates an SQL update module for each propagation request belonging to a generalized mapping case, except for propagation requests defined with a mapping direction of RH. The SQL update module uses the propagation request ID as a load module name.

USERID (OPTIONAL)

Identifies the owner of the propagation request. Specify an identifier of your choice for *userid*.

The default is no user ID.

NODEID (OPTIONAL)

Identifies the node ID of the owner of the propagation request. Specify an identifier of your choice for *nodeid*.

The default is no node ID.

PRTYPE (E | L | F | U) (REQUIRED unless //MVGPARM provides a default value)

Identifies the type of propagation request you want to create:

E Extended-function
L Limited function
F Full-function
U User mapping

The majority of IMS segments in an installation can be propagated using the generalized mapping logic of types E, L and F. If you want to perform DB2-to-IMS synchronous propagation and have the support of the generalized mapping logic of IMS DPROP, you must code PRTYPE=E.

Refer to topic page 51 for more information on PRTYPE.

MAPCASE (1 | 2 | 3 |) (REQUIRED for a generalized mapping case propagation request unless //MVGPARM provides a default value)

For generalized mapping cases, this field identifies to which mapping case your propagation request belongs, 1, 2 or 3.

Mapping case 1 provides mapping between IMS data of one entity segment and its concatenated key to or from one relational table.

Mapping case 2 provides mapping between IMS data of one entity segment and its concatenated key, and one or more extension segments, to or from one relational table.

Mapping case 3 provides mapping between IMS data of one internal segment and its concatenated key to or from one relational table.

If you selected user mapping, this field has no effect in IMS DPROP however, it must be blank, 1, 2, or 3.

MAPDIR (HR | TW | RH) (REQUIRED unless //MVGPARM provides a default value)

Indicates the mapping direction.

HR indicates that data is propagated in a hierarchical-to-relational only direction.

TW indicates that data is synchronously propagated in both a hierarchical-to-relational direction and in a relational-to-hierarchical direction.

RH indicates that data is synchronously propagated in a relational-to-hierarchical only direction.

TW or RH cannot be specified for propagation requests defined with PRTYPE=L and PRTYPE=F. For asynchronous propagation and user asynchronous propagation, you cannot submit propagation requests with a mapping direction of TW and RH.

TABQUAL2 (OPTIONAL)

Is a table qualifier used for validation purposes only.

Provide a value in the TABQUAL2 column only if you specify an unqualified table name for the propagated table; that is, if the content of the TABQUAL column in the TAB MVG input table is blank.

If you specify an unqualified table name for the propagated table, IMS DPROP must find the description of a “model table” in the DB2 catalog. The model table should have the same attributes as the propagated table (the model table can be, but does not have to be, the table being propagated). IMS DPROP generates mapping information in the PR based on the DB2 catalog description of the model table.

You can specify the table name qualifier of the model table in the optional TABQUAL2 column (it must be a valid DB2 authorization ID). If you do not provide a TABQUAL2 value, then the userid of the MVGU job is used as default qualifier of the model table.

The name of the model table consists of:

- The qualifier specified on the TABQUAL2 column (or the userid of the MVGU job)
- The table name specified in the TABNAME column of the TAB MVG input table

If you specify an unqualified table name for the propagated table, IMS DPROP assumes that the model table has been created and has the same attributes as the table propagated by the propagation request.

ERROPT (BACKOUT | IGNORE) (REQUIRED unless //MVGPARM provides a default value)

Indicates the propagation request error option.

If propagation cannot be successfully completed, this column tells the following to either BACKOUT the changes since the last commit point or IGNORE the error:

- RUP
- HUP

When you select ERROPT=IGNORE, IMS DPROP does not abend the application; instead, it writes diagnosis information and returns to the application program without indicating errors. While this reduces failures that could impact

existing applications, it results in inconsistencies between the IMS and DB2 data. Because of this, you might eventually change to ERROPT=BACKOUT. You can change ERROPT with the Status Change utility (SCU), as described in Chapter 8, “Status Change Utility (SCU) for LOG-ASYNC ,” on page 213.

MAXERROR (OPTIONAL)

Identifies the maximum number of propagation failures that are:

- Reported for this propagation request to OS/VS consoles and to the audit trail within a 15-minute interval.
- Documented for this propagation request with detailed trace records in the IMS DPROP trace within a 15-minute interval.

MAXERROR applies only to a propagation request with ERROPT=IGNORE and limits only the reporting of propagation failures that do not result in a backout.

When the MAXERROR limit has been exceeded, propagation continues without error messages being written.

This column limits only those messages written to the OS/VS console and to the audit trail. It also limits detailed trace information. It does not limit messages written to the optional //EKYPRINT data set and to the IMS DPROP trace.

Valid values are between 0 and 32767. The default of 0 means *no limit*.

ACTION (ADD | REPL) (OPTIONAL)

Indicates the action: add a new propagation request (ADD) or replace a propagation request (REPL). ADD is the default.

When you select ADD, IMS DPROP first checks to see that the propagation request does not already exist in the mapping tables. If it does not, IMS DPROP creates a new propagation request in the IMS DPROP directory and the SQL module libraries. Otherwise, IMS DPROP issues an error message and returns a nonzero return code to the MVGU.

When you select REPL, IMS DPROP either replaces the already existing propagation request, or, if the propagation request does not exist on the mapping tables, it adds the propagation request.

For synchronous propagation, if you replace an existing propagation request, either it must be inactive or the IMS DPROP system must be emergency stopped.

PRSET (OPTIONAL)

Identifies the set of propagation requests in which this PR belongs. If you do not provide a set name, then the name defined at IMS DPROP generation is used as a default. The *prset* name must be alphanumeric.

Refer to the appropriate Administrators Guide for your propagation mode for a more detailed description of the PRSET.

PROPSUP (Y |N) (OPTIONAL)

Indicates selective suppression of synchronous propagation.

Select N if you do not want to allow Segment exit routines to return a code of 8 during data propagation. When Segment exit routines return with a return code of 8, it is considered an error, and IMS DPROP issues an abend.

If you select Y and a Segment exit routine issues a return code of 8, it is not considered an error; instead, synchronous propagation for this propagation request is suppressed.

Selectively suppressing the synchronous propagation of IMS and DB2 data changes results in inconsistencies between both data copies. These inconsistencies can eventually result in synchronous propagation failures of other IMS or DB2 updates.

If you choose Y and you do not specify the USE command of the CCU (refer to Chapter 8, “Status Change Utility (SCU) for LOG-ASYNC ,” on page 213 for more information on this command), CCU executions could report inconsistencies that do not exist.

EXITNAME (REQUIRED for user mapping unless //MVGPARM provides a default)

Identifies the Propagation exit routine to be invoked for user mapping (if PRTYPE=U). For generalized mapping, leave this field blank.

PROCSED (Y |not Y) (OPTIONAL)

Indicates whether the propagation request has (Y) or has not (not Y) been processed by MVGU.

After the MVGU processes the propagation request, the source PR is not deleted from the MVG input tables. Instead, the MVGU marks the propagation request as processed by setting the PROCSED flag to Y. Any value other than Y in this column indicates that the PR has not yet been processed by the MVGU.

To reprocess a propagation request, you must set PROCSED to a value other than Y before running the MVGU again.

KEYORDER (ASC | DSC | ANY) (OPTIONAL)

Indicates the column sequence of the primary key index of the propagated table.

The default is ASC.

This field indicates if all columns of the primary key of the propagated table are in ascending (ASC) or descending (DSC) sequence, or if MVG must access the DB2 catalog to get the defined ordering sequence of each column of the primary key of the propagated table (ANY= any sequence). When you specify ANY, IMS DPROP’S access to the DB2 catalog is typically lengthy and inefficient.

If you know that all primary key columns have been defined in the DB2 index as ASC or DSC, define them here as ASC or DSC. This improves the performance of MVGU processing.

If you incorrectly specify ASC or DSC, the CCU will report errors where none likely exists.

BIND (OPTIONAL)

Contains BIND options for the SQL update module. The default is no BIND options.

If you want IMS DPROP to automatically bind the PACKAGE of the SQL update module, you must specify bind options, including the collection ID where this package is to be bound.

You can provide these bind options either in the BIND column of the PR MVG input table, or in the propagation parameter default data set (//MVGPARM) in the form of a string of DB2 specifications, up to a maximum of 254 bytes, without a continuation character. An example of this is shown in Figure 40 on page 96.

IMS DPROP ensures that the following keywords, which are not supported, are not specified in the string:

- MEMBER (IMS DPROP generates this operand)
- BIND
- PLAN
- COPY
- LIBRARY

If the collection ID is not specified in the PACKAGE keyword, the other bind options are ignored and no package is created.

For performance reasons, specify VALIDATE(BIND) rather than the DB2 default of VALIDATE(RUN). This prevents DB2 from performing a package validation each time the RUP calls an SQL update module.

If you have specified an unqualified propagated table name propagation request, you should use the QUALIFIER operand to set the correct qualifier.

Figure 40 shows an example of BIND options:

```
PACKAGE(ONLINE)
OWNER(DBADMIN)
QUALIFIER(DB2PROD)
VALIDATE(BIND)
```

Figure 40. Example of BIND options for the package of the SQL update module

PATH (ID | DENORM |) (REQUIRED for path data unless //MVGPARM provides a default)

Identifies the kind of non-key path data selected in the propagation request. The default is no path data selected.

When not blank, this propagation parameter indicates that path data is selected in the propagation request and describes the kind of path data:

ID Specifies that you include only IMS ID fields as path data, that is, fields which cannot change their value.

DENORM

Specifies that you include fields that can change their value.

For propagation requests with PRTYPE=E, you cannot specify PATH=DENORM.

AVU (Y | N |) (OPTIONAL)

Avoids unnecessary updates. With this parameter, you can influence the performance of IMS-to-DB2 propagation. This parameter specifies whether or not IMS DPROP should determine, during the replacement of a segment, whether at least one propagated field has changed.

Y The replacement of an IMS segment results in a propagating SQL update, only if at least one propagated field has changed.

AVU=Y can reduce the total IMS DPROP path length when only a subset of the fields of a segment is propagated.

N The replacement of an IMS segment results in a propagating SQL update, even if no propagated field has changed.

AVU=N can reduce the total IMS DPROP path length when, for example, all fields of a segment are propagated. When AVU=N, IMS DPROP does not compare each field to find if at least one of them has changed.

If you do not specify AVU, IMS DPROP determines a default setting at propagation time.

- AVU=Y is the default value:
  - When propagating internal segments with mapping case 3 propagation requests, or
  - When propagating IMS segments with mapping case 1 and 2 propagation requests, and at least one non-key byte of the segment is not propagated.
- AVU=N is the default value in all other cases.

DEFVEXT (Y | N |) (OPTIONAL)
Identifies how IMS DPROP handles extension segments propagated exclusively from default values and NULLS during DB2-to-IMS synchronous propagation.

This parameter is only used for mapping case 2 propagation requests performing DB2-to-IMS synchronous propagation. It tells IMS DPROP what to do with an extension segment during synchronous propagation of an SQL insert or update when:

- All the target columns of an extension segment are either null or contain default values, and
- The SQL insert or update changes at least one column mapped to the extension segment.

N The synchronous propagation of the SQL insert or update results in zero occurrences of the extension segment type.

Y The synchronous propagation of the SQL insert or update results in an occurrence of the extension segment type, even if all of its fields are propagated from columns having a NULL or default value.

The default is DEFVEXT=Y or blank.

COMMENT (OPTIONAL)
Comments used to document the propagation request.

These comments are stored in the propagation request mapping table.

The WHR MVG Input Table (DPRIWHR)

With the generalized mapping logic of IMS DPROP, you can optionally include a WHERE clause in the propagation request definition.

The WHERE clause is used to propagate data selectively. Use the WHERE clause to specify under which conditions an occurrence of the segment type should be propagated by the propagation request from IMS to DB2. For example, you can set a condition based on field values or a combination of field values.

IMS DPROP supports WHERE clauses for propagation requests belonging to mapping case 1 and 2. WHERE clauses are not supported for mapping case 3

propagation requests. For more information on using the WHERE clause for DB2-to-IMS synchronous propagation, refer to the appropriate Administrators Guide for your propagation mode.

You specify the optional WHERE clause in rows of the WHR table.

If the propagation request does not have a WHERE clause, do not provide WHR rows. If the propagation request has a WHERE clause, provide one or more WHR rows. Describe in the WHR rows the conditions that the IMS data must satisfy to be propagated by IMS DPROP for a generalized mapping case.

If your WHERE text is longer than 254 bytes, provide several rows to specify the text, with an ascending sequence number to enforce the uniqueness of the DB2 primary key of the table.

The columns in the WHR table are described in Table 8.

Table 8. Columns of the WHR MVG Input Table and Their Data Types

Column	Data type	Description
PRID	char(8)	Propagation request ID
SEQUENCE	smallint	Sequence number
WHRTEXT	varchar(254)	at most 254 bytes of the WHERE text

PRID (REQUIRED)
Is the propagation request identifier. It must be the same value specified in the PRID column of the related propagation request row.

SEQUENCE (REQUIRED)
Is the sequence number used to enforce the uniqueness of the DB2 primary key of this table when the WHERE text greater is than 254 bytes and requires more than one row.

WHRTEXT (REQUIRED)
Contains one portion (up to 254 bytes) of the WHERE text.

Syntax of the WHERE clause

The syntax of the WHERE clause is shown in Figure 41. *operand1* and *operand2* can be field names or constants.

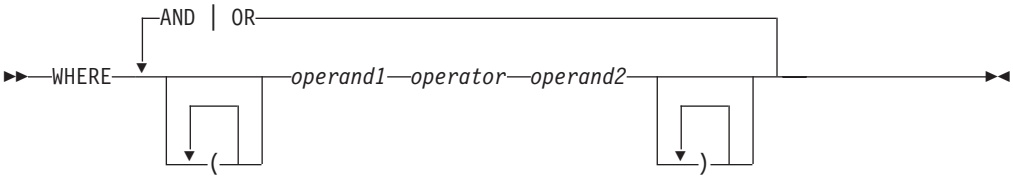

Figure 41. The WHERE Clause

Examples of the WHERE Clause

Following are examples of how the WHERE clause can be written.

Note: The operators *AND* and *OR* are not mutually exclusive.

```
WHERE FIELD1 > FIELD2
WHERE FIELD1 => 'ABC'
WHERE FIELD1 => 'ABC' AND FIELD2 = 'CDE'
WHERE ((FIELD1 => 'ABC') AND (FIELD2 = 'CDE')) OR (FIELD4 = 4)
```

Figure 42. Examples of WHERE clauses

Naming the Fields

For TYPE=E propagation requests, all fields in the WHERE clause must be selected for propagation. For other types of propagation requests, fields in the WHERE clause do not have to be selected for propagation.

In order to understand which fields can be included in the WHERE clause, refer to the appropriate Administrators Guide for your propagation mode.

The remainder of the WHERE clause, as described in the WHR table, is identical to its description in the DataRefresher EXTRACT statement. The following list directs you to the information on the remaining WHERE clause descriptions in the DataRefresher EXTRACT statement.

- “Forming the Conditions ” on page 71
- “Specifying Multiple Conditions” on page 71
- “Restrictions for Numerical Operands ” on page 72
- “Rules for Special Cases ” on page 72
- “WHERE Clause Constants ” on page 73

The SEG MVG Input Table (DPRISEG)

In the SEG table, you specify the segments participating in propagation. Your definition of the segments will vary depending on the mapping case you choose.

IMS Segments and Internal Segments

Depending on the mapping case for the propagation request being coded, you must provide SEG rows either for IMS segments only or for both IMS and internal segments.

When you describe an IMS segment, you do not have to provide information such as parent segment name, segment length, format, key specifications, database organization. MVG obtains this information for you when it accesses the DBD.

Internal segments are not IMS segments. They describe embedded structures located within one IMS segment type. A typical example of an embedded structure is a *repeating group* of fields with a fixed or variable number of occurrences within the IMS segment.

Internal segments can be propagated by mapping case 3 propagation requests or by user mapping. Refer to the appropriate Administrators Guide for your propagation mode for a description of IMS DPROP support for the propagation of internal segments with mapping case 3 propagation requests.

Containing IMS Segment

The IMS segment that contains one or more internal segment types is considered to be the hierarchical parent of the internal segments. This IMS segment is referred to as the *containing IMS segment*.

Segment Exit Routines and Internal Segments

As described in the appropriate Administrators Guide for your propagation mode, propagation of internal segments with the generalized mapping logic of IMS DPROP often requires that the containing IMS segment be processed by a Segment exit routine.

For example, propagation of internal segments with Propagation requests having PRTYPE=E requires that you provide a Segment exit routine; IMS DPROP ensures that you identify the Segment exit routine on the SEG row of the containing IMS segment, not on the SEG row describing the internal segment.

Segment exit routines always process an entire IMS segment (as opposed to an individual internal segment).

Nesting Internal Segments

The generalized mapping logic of IMS DPROP does not support the nesting of internal segments; that is, an internal segment containing *other* internal segments. Propagation of nested internal segments can be done with user mapping (TYPE=U PRs).

Providing SEG Rows Depending on the Mapping Case

When you define a SEG row, you must specify the role that the described segment plays in the propagation request. It can be one of the following:

E The entity segment. It can be an IMS segment or an internal segment.

X An extension segment of the entity segment. This can only be an IMS segment.

P A parent or ancestor of the entity segment. This can only be an IMS segment.

C The containing IMS segment, if the entity segment is an internal segment.

S A STARTSEG segment; that is, an internal segment used to locate the entity segment. See the description of the column ROLE in “SEG Row for an Internal Segment” on page 103 for a detailed description of STARTSEG segments.

SEG Rows for Mapping Case 1

If you are defining a propagation request belonging to generalized mapping case 1, you need to insert:

- One row for the entity segment (ROLE=E)
- One row for its physical parent (ROLE=P), and
- One row for each physical ancestor up to the root (ROLE=P)

SEG Rows for Mapping Case 2

If you are defining a propagation request belonging to generalized mapping case 2, you need to insert:

- One row for the entity segment (ROLE=E)
- One row for each extension segment ROLE=X)
- One row for the physical parent (ROLE=P), and

- One row for each physical ancestor of the entity segment up to the root (ROLE=P)

SEG Rows for Mapping Case 3

If you are defining a propagation request belonging to generalized mapping case 3, you need to insert:

- One row for the entity segment, which is an internal segment (ROLE=E)
- One row for the containing IMS segment (ROLE=C), and
- One row for each physical parent/ancestor of the containing IMS segment up to the root (ROLE=P)
- Additionally, you might need to insert SEG rows for STARTSEG segments (ROLE=S)

SEG Rows for User Mapping

If you are defining a propagation request belonging to a user mapping case, you need to insert one row for each segment type that is being propagated by the propagation request being defined.

When processing a DL/I update of one of these segment types, the RUP calls the Propagation exit routine identified in the EXITNAME column of the propagation request table.

If there are multiple propagation requests belonging to a user mapping case, and each has a SEG row for a particular segment type, the RUP processes the multiple propagation requests and issues one call for each propagation request to the Propagation exit routine.

For DB2-to-IMS synchronous propagation, the Propagation exit routine is called whenever one of the tables identified in the DPRITAB rows is updated.

Columns of the SEG Table

Table 9 describes the columns of the SEG MVG input table. Some columns are only required for internal segments. “SEG Row for an IMS Segment ” on page 102 and “SEG Row for an Internal Segment” on page 103 describe how to code the SEG row depending on whether you describe an IMS segment or an internal segment.

Table 9. Columns of the SEG MVG Input Table and Their Data Types

Column	Data type	Description
PRID	char(8)	Propagation request ID
DBNAME	char(8)	Database name
SEGNAME	char(8)	Segment name
SEGEXIT	char(8)	Segment exit name, if any
SEGEXITL	integer	Segment length output of segment exit, if SEGEXIT is coded
SEGEXITF	char(2)	Segment format output of segment exit, if SEGEXIT is coded
ROLE	char(2)	Segment role (E|X|P|C|S)
PCBLABEL	char(8)	Name of the IMS PCB used to perform DB2-to-IMS synchronous propagation.
FORMAT	char(2)	Segment format if internal segment (FI/VI)
OCCURS	char(32)	Number of occurrences of the internal segment
START	char(38)	Start position of the internal segment

Table 9. Columns of the SEG MVG Input Table and Their Data Types (continued)

Column	Data type	Description
BYTES	integer	Length in bytes of the internal segment
NEXT	char(38)	Next occurrence of the internal segment

SEG Row for an IMS Segment

The following columns are applicable when you describe an IMS segment:

PRID (REQUIRED)
Is the propagation request identifier. It must be the same value specified in the PRID column of the related propagation request row.

DBNAME (REQUIRED)
Identifies the database participating in propagation. You provide the name of the physical IMS DBD that must exist in the IMS DBD library.

SEGNAME (REQUIRED)
Identifies the segment participating in propagation.

This is the IMS name of the segment type used for the segment in the physical DBD generation. SDEPs of DEDBs are only supported for propagation requests with user mapping (PRTYPE=U).

SEGEXIT (OPTIONAL)
Identifies the Segment exit routine to be called by the RUP or, for synchronous propagation, the HUP to transform the segment, if necessary.

Leave SEGEXIT blank if the segment should not be processed by a Segment exit routine. Otherwise, specify the name of the Segment exit routine.

Note: IMS segments containing internal segments must often be processed by a Segment exit routine. This is required when such IMS segments are propagated by TYPE=E propagation requests. Identify the Segment exit routine in the SEGEXIT column of the SEG row describing the IMS segment, not in the SEG row describing the internal segment. You must identify the Segment exit routine when defining both:
- The mapping case 3 propagation request that propagates the internal segment
- The mapping case 1 or 2 propagation request that propagates the containing segment

SEGEXITL (OPTIONAL)
Indicates the length of the segment in its IMS DPROP format if SEGEXIT is coded.

If you select a SEGEXIT, this column indicates the length of the segment:
- After it is processed by the Segment exit routine, for IMS-to-DB2 propagation
- Before it is processed by the Segment exit routine, for DB2-to-IMS synchronous propagation

For variable-length segments, specify the maximum length in bytes. The minimum length is 3, which includes 2 for the 2-byte length field located at the start of the segment.

The length can be any integer from 1 to 32760.

SEGEXITF (F | V) (OPTIONAL)

If you select a SEGEXIT, this column indicates the format of the segment as it appears:

- After it is processed by the Segment exit routine, for IMS-to-DB2 propagation
- Before it is processed by the Segment exit routine, for DB2-to-IMS synchronous propagation

Specify F for a fixed-length segment format and specify V for a variable-length segment format.

ROLE (E | X | P | C) (REQUIRED)

Indicates the role the segment plays in the propagation request. It can be:

E The entity segment itself

X An extension segment of the entity segment (mapping case 2)

P A parent or ancestor of the entity segment

C The IMS segment containing the entity segment (mapping case 3)

For user mapping, this field has no effect in IMS DPROP however, it must be blank or contain one of the preceding values.

As the GUU cannot always determine the role of the segments in user mapping propagation requests, it provides segment-level sensitivity for all segments (including parent and ancestor segments).

If PATH data is not propagated, segment level sensitivity is not required for the parent and ancestor segments and you should remove the parent and ancestor segments from the GUU output.

PCBLABEL (REQUIRED depending on mapping direction and segment role)

If the propagation request is a generalized mapping case and has been defined with MAPDIR=TW or MAPDIR=RH, this column must contain the name of the PCB used to perform DB2-to-IMS synchronous propagation.

For propagation requests defined with mapping direction TW or RH, provide a PCBLABEL for the SEG row describing:

- The entity segment, when the propagation request is mapping case 1 or 2
- The containing IMS segment, when the propagation request is mapping case 3

For user mapping, this field will be moved, unvalidated, to the IMS DPROP directory.

The remaining columns are for internal segments. If you provide any value in these columns for an IMS segment (when FORMAT remains blank), IMS DPROP considers this an error.

SEG Row for an Internal Segment

The following description applies to columns used to describe an internal segment. Remember that for a propagation request with mapping case 3, you also have to describe the containing IMS segment of the entity segment, as well as the containing segment's physical parent and physical ancestors. When defining internal segments, you need to describe:

- Whether the internal segment has a fixed or variable number of occurrences within the containing IMS segment
- Where the first occurrence of the internal segment starts within the containing IMS segment

- Whether the internal segment has a fixed or variable length
- How IMS DPROP and DataRefresher can step from one occurrence of the internal segment to the next occurrence (if the internal segment has a variable length)

This is explained in detail below:

- **Does the internal segment have either a fixed or a variable number of occurrences within the containing IMS segment?**

  For a fixed number of occurrences, specify the value in the OCCURS column.

  For a variable number of occurrences, specify a *fieldname* in the OCCURS column. *fieldname* is the name of a count field located in the containing IMS segment. The count field contains the actual number of occurrences.

  For propagation with TYPE=E propagation requests (they support DB2-to-IMS synchronous propagation), IMS DPROP requires that the internal segment be defined as having a variable number of occurrences recorded in a count field. For TYPE=E propagation requests, you cannot propagate the count field.

  If you need to propagate internal segments having a variable number of occurrences, but the IMS segment does not have a count field, consider using a Segment exit routine. It can edit the IMS segment so that the edited format has a count field.

- **Where is the start of the first occurrence of an internal segment within its containing IMS segment?**

  You need to define this to IMS DPROP using the START column.

  You can use one of the three following forms in the START column:

  n
  : Use a numeric value, *n*, when the first occurrence of the internal segment has a fixed start position within its containing IMS segment. *n* identifies the fixed start position.

  fieldname+n
  : Use *fieldname+n* when the first occurrence of the internal segment starts *n* bytes from the end of the named field. The named field must be a field located in the containing IMS segment.

    If the named field has a variable length, then the first internal segment has a variable start position.

  segname+n
  : Use *segname+n* when the IMS segment contains multiple internal segment types. When specifying START=*segname+n*, the first occurrence of the internal segment starts *n* bytes from the last occurrence of the other internal segment identified by segname.

- **Does the internal segment have a fixed or variable length?**

  You define this on the FORMAT column.

- **How can IMS DPROP and DataRefresher step from one occurrence of the internal segment to the next?**

  For fixed-length internal segments, defining the length on the BYTES column is the only information IMS DPROP needs to step from one occurrence to the next.

  For variable-length internal segments, specifying *fieldname+n* in the NEXT column indicates that the next occurrence of the internal segment starts *n* bytes from the end of the named field. The named field must be located in the internal segment being described.

  If the named field has a variable length, then the internal segment also has a variable length.

Recommendations for Describing Internal Segments

1. For DB2-to-IMS synchronous propagation, avoid redefining the same byte of the IMS segment so that it is part of multiple propagated fields. For example, avoid definitions resulting in one byte becoming part of:
   - Two fields located in two different internal segments, or
   - Not only one field located in an internal segment, but also another field located in the containing segment.

   Ignoring this recommendation can result in data inconsistencies.
2. If you describe the same IMS segment in multiple propagation requests, be sure to consistently define the internal segments and the containing IMS segment in all propagation requests. For example:
   - Use the same fixed or variable formats, the same lengths, the same fixed or variable start positions, the same segment names.
   - Use different names for different internal segments.

   Ignoring this recommendation can result in data inconsistencies.

The following columns are applicable when you describe an internal segment:

PRID (REQUIRED)
Is the propagation request identifier. It must be the same value specified in the PRID column of the related propagation request row.

DBNAME (REQUIRED)
Identifies the database participating in propagation. You provide the name of the physical IMS DBD that must exist in the IMS DBD library.

SEGNAME (REQUIRED)
Identifies the segment participating in propagation. The identified segment is an internal segment.

SEGEXIT (NOT ALLOWED)
Identifies the Segment exit routine to be called by the RUP or, for synchronous propagation, the HUP to transform the segment, if necessary.

Leave this column blank for internal segments. If your internal segment requires a segment exit routine, provide this routine in the SEG row describing the containing IMS segment, not in this SEG row.

SEGEXITL (NOT ALLOWED)
Leave this column blank for an internal segment.

SEGEXITF (NOT ALLOWED)
Leave this column blank for an internal segment.

ROLE (E | S) (REQUIRED)
Indicates the role the internal segment plays in the propagation request. It can be:

E The entity segment of a mapping case 3 propagation request. It is an internal segment.

S A STARTSEG segment. A STARTSEG segment is an internal segment that has the same containing IMS segment as the entity segment and that physically precedes the entity segment.

When you locate the entity segment by coding *segname+n* in the column START, you have to insert a SEG row with ROLE=S for the segment named in *segname*.

For user mapping, this field has no effect in IMS DPROP however, it must be blank or contain one of the above values.

As the GUU cannot always determine the role of the segments in user mapping propagation requests, it provides segment-level sensitivity for all segments (including parent and ancestor segments).

If PATH data is not propagated, segment level sensitivity is not required for the parent and ancestor segments and you should remove the parent and ancestor segments from the GUU output.

PCBLABEL (NOT ALLOWED)
Leave this column blank for an internal segment.

If the described propagation request performs DB2-to-IMS synchronous propagation, provide a PCBLABEL in the SEG row describing the containing IMS segment, not in this SEG row.

FORMAT (FI | VI) (REQUIRED)
Specifies the segment format.

Specify FI or VI in this column to describe an internal segment. Specify FI for a fixed-length internal segment or VI for a variable-length internal segment.

If you do not specify any value in this column, IMS DPROP assumes that your described segment is an IMS segment and therefore does not validate the next columns.

OCCURS (REQUIRED)
Tells IMS DPROP how many occurrences of this internal segment type there are in the containing IMS segment. OCCURS can be an integer or the name of a counter field in the containing IMS segment, whose value is the number of occurrences.

Specify:

n For a fixed number of occurrences, replace *n* with that number.

fieldname
For a variable number of occurrences, replace *fieldname* with the name of the field that contains the value for the number of occurrences. This count field must:
- Be located in the containing IMS segment
- Physically precede the start of this described internal segment
- Be a positive integer

For TYPE=E propagation requests, you should not propagate the count field. To propagate an internal segment with a TYPE=E propagation request, IMS DPROP requires that the internal segment be defined as having a variable number of occurrences and that you specify a *fieldname* in the OCCURS column.

START (REQUIRED)
Specifies the starting position of the first occurrence of the internal segment type within the containing IMS segment.

If the containing IMS segment is processed by a Segment exit routine, your definition of the start position should apply to the containing IMS segment as described to IMS DPROP. For IMS-to-DB2 propagation, the description should apply to the segment format *after* processing by the Segment exit routine.

For fixed starting position, use a positive integer to specify the starting position of the first occurrence of the internal segment within the containing segment. The first byte within a segment is considered to have the start position of 1 (not 0). If the containing IMS segment has a variable-length format, be sure to account for the 2-byte length field at the start of the segment. For example, a value of 1 means that the internal segment begins at the start of the length field of the containing IMS segment, not at the byte following the length field.

For variable starting position, code one of the following:

fieldname+n

Replace *fieldname* with the name of a field defined in the containing IMS segment.

Replace *n* with the number of bytes from the **end** of the named field to the starting position of the first occurrence of this internal segment. *n* must be a positive integer.

If the named field and this internal segment type are adjacent, code *fieldname*+1.

If the named field has a variable length, then the first occurrence of the internal segment type has a variable start position.

segname+n

Use *segname+n* if the containing IMS segment contains multiple internal segment types and you currently describe the nth (n>1) internal segment type.

Replace *segname* with the name of another internal segment type (a STARTSEG segment) defined in the same containing IMS segment and preceding the internal segment just described.

Remember to define this STARTSEG segment by providing an additional SEG row, with a ROLE of S.

Replace *n* with the number of bytes from the **end** of the last occurrence of the named STARTSEG segment to the starting position of the first occurrence of this internal segment. *n* must be a positive integer.

If the two internal segment types are adjacent, code *segname*+1.

BYTES (NOT ALLOWED for variable-length segments REQUIRED for fixed-length segments unless NEXT is provided)

Indicates the length in bytes of the internal segment.

If you do not provide the segment length in this column, you must provide a value in NEXT.

For variable-length segments, leave this column zero. For fixed-length segments, you can specify either BYTES or NEXT.

NEXT (REQUIRED for variable-length segments REQUIRED for fixed-length segments unless BYTES is provided)

Tells IMS DPROP how to step from one occurrence of the internal segment to the next one.

For fixed length internal segments, you can specify either BYTES or NEXT. For a variable-length segment, you must specify NEXT.

Code the following in the column NEXT:

fieldname+n

Replace *fieldname* with the name of a field located in the internal segment.

Replace *n* with the number of bytes from the end of the named field to the next occurrence of the internal segment. The value of *n* must be a positive integer.

If the named field and the next occurrence of the internal segment are adjacent, code *fieldname*+1. If the named field has a variable length, then the internal segment has a variable length.

For variable-length segments, IMS DPROP expects that each occurrence of a variable-length internal segment be large enough to contain the position identified by (*fieldname + n*) - 1.

The TAB MVG Input Table (DPRITAB)

In the TAB table, specify the tables that are the propagated tables for this propagation request.

If you are using generalized mapping case 1, 2, or 3, you must specify *only one* table for each propagation request.

If you are using your own mapping cases (PRTYPE=U), you can specify more than one table. You must specify at least one table. For DB2-to-IMS synchronous propagation, the Propagation exit routine identified in the DPRIPR row is called whenever one of these tables is updated.

You must create the propagated or “model” tables before the MVGU creates your propagation request based on the information stored in the MVG input tables. For generalized mapping cases, the propagated tables must have a primary key.

The columns in the TAB table are described in Table 10.

Table 10. Columns of the TAB MVG Input Table and Their Data Types

Column	Data type	Description
PRID	char(8)	Propagation request ID
TABQUAL	char(8)	Qualifier of the propagated table
TABNAME	varchar(18)	Name of the propagated table

PRID (REQUIRED)

Is the propagation request identifier. It must be the same value specified in the PRID column of the related propagation request. row.

TABQUAL (OPTIONAL)

Is the qualifier of the propagated table.

Provide in this column either the table name qualifier of the propagated table or blanks.

PR definitions with a table name qualifier are typically used for production environments. They support propagation to or from only one table: the table identified by the content of the TABQUAL and TABNAME columns.

PR definitions without a table name qualifier (with blanks in TABQUAL) can be used in some test environments. They support propagation to or from one of multiple, identically structured tables having the unqualified table name in the TABNAME column.

For more information on propagation requests defined with and without table name qualifiers, refer to the appropriate Administrators Guide for your propagation mode.

TABNAME (REQUIRED)
Identifies the name of the table to or from which data is propagated. You must enter a value in this field.

For a generalized mapping case propagation request, the table name must be unique across all propagation requests on the mapping tables of the IMS DPROP directory; that is, the table can only be propagated by one propagation request. A more detailed description of this rule follows:

- If the same table name is specified once on one propagation request. with a qualifier and another time on another propagation request. without a qualifier, IMS DPROP considers that both propagation requests propagate the same table. IMS DPROP considers this an error.
- If the same table name is specified without qualifiers on two different propagation requests, IMS DPROP considers that both propagation requests propagate the same table. IMS DPROP considers this an error.
- If the same table name is specified on two different Propagation requests with two different qualifiers identifying two different propagated tables, no error exists.

For user mapping cases, several propagation requests can propagate data to or from the same table. However, tables propagated by a generalized mapping case propagation request cannot also be propagated by a user mapping case propagation request; likewise, tables propagated by user mapping case propagation request. cannot also be propagated by a generalized mapping case propagation request.

For DB2-to-IMS and two-way synchronous propagation, you cannot specify the same table name in one propagation request as qualified, and later in another propagation request. specify it as unqualified. You must specify a specific table name either always *with* a qualifier, or always *without* a qualifier.

The FLD MVG Input Table (DPRIFLD)

When you have identified your IMS segments and your propagated DB2 tables, you must specify the fields participating in propagation and the columns to which they correspond. You code this information on the FLD table.

Propagated and Nonpropagated Fields

You must insert one FLD row for each field propagated or included in the optional WHERE clause. You can also insert one row for IMS fields that are *not* propagated.

Specifying fields that are not propagated is recommended for documentation purposes if the propagation request propagates data only from IMS to DB2.

If the propagation request is subject to DB2-to-IMS synchronous propagation (PRTYPE=E), specify IMS fields that are not propagated. This ensures that the HUP correctly initializes these fields when inserting an IMS segment or when

extending a variable-length IMS segment. For example, instead of becoming X'00', numeric fields will become zero, and character fields will become blank.

Variable-length Fields

For variable-length IMS fields, insert two rows into the FLD table: one row describing the variable-length field itself, and one row describing a field that contains the length of the variable-length field. The field is called the length field. In the row describing the variable-length field, the LENFIELD column must contain the name of the length field. In the row describing the length field, set the COLNAME column to blanks in the cases where the length field is not propagated.

Using a Field Exit

For fields that are transformed by a Field exit routine into a variable-length format, do *not* identify the name of a length field in the LENFIELD column. For such fields, the length must be provided by the Field exit routine.

Using a Segment Exit

If a segment is processed by a Segment exit routine, the FLD rows describe the segment fields as they appear in the IMS DPROP format of the segment.

For IMS-to-DB2 propagation, this is the segment format *after* it is processed by the segment exit routine. For DB2-to-IMS synchronous propagation, it is the segment exit format *before* it is processed by the segment exit routine.

Describing Special Fields of a Containing IMS Segment

If you describe a containing IMS segment, you might need to provide FLD rows for the field containing the occurrence of the internal segment and for the field used to determine the start position of the internal segment, if necessary.

Table 11 describes some rules for selecting fields.

Table 11. Rules for Describing Fields for the FLD MVG Input Table

For a propagation request belonging to mapping case...	Describe fields...
1	• Located in the entity segment • Located in the IMS key of the parent/ancestors of the entity segment • Located in the data portion of the physical parent/ancestors of the entity segment if path data is selected • Used in the optional WHERE clause, if these fields do not belong to the cases above
2	• Located in the extension segments • Located in the entity segment • Located in the IMS key of the parent/ancestors of the entity segment • Located in the data portion of the physical parent/ancestors of the entity segment if path data is selected • Used in the optional WHERE clause, if these fields do not belong to the cases above

Table 11. Rules for Describing Fields for the FLD MVG Input Table (continued)

For a propagation request belonging to mapping case...	Describe fields...
3	• Located in the entity segment, which is an internal segment • Located in the IMS key of the containing IMS segment (the segment that contains the internal segment) • Located in the IMS key of the parent/ancestors of the containing IMS segment • Located in the data portion of the containing IMS segment and of the physical parent/ancestors of the containing IMS segment if path data is selected
User	(IMS DPROP does not require that you insert rows in the FLD table for user mapping cases)

IMS DPROP has some rules for the mapping of key fields. Ideally, you should map all fields that are within the fully concatenated key of an entity segment to the columns of a DB2 primary key. Refer to the appropriate Administrators Guide for your propagation mode for information on mapping rules for keys.

All columns of the DB2 primary key of the target table must be propagated from the IMS fields. Therefore, you should insert one FLD row for each column of the DB2 primary key. The same is true for columns that are defined to DB2 as NOT NULL.

Columns mapped from fields located in extension segments of a mapping case 2 propagation request should be defined to DB2 as NOT NULL WITH DEFAULT or nullable. Columns mapped from fields located in variable-length segments that could be missing in some segment occurrences should also be defined to DB2 as NOT NULL WITH DEFAULT or nullable.

For a propagation request belonging to a generalized mapping case, IMS DPROP checks that a column is mapped to by only one field.

The columns in the FLD table are described Table 12.

Table 12. Columns of the FLD MVG Input Table and Their Data Types

Column	Data type	Description
PRID	char(8)	Propagation request ID
DBNAME	char(8)	Database name
SEGNAME	char(8)	Segment name
FLDNAME	char(32)	Field name
POSITION	integer	Field start position in segment
DATATYPE	char(2)	Data type of the IMS field (IMS DPROP terminology)
DATATYP2	char(8)	Data type of the IMS field (DB2 terminology)
BYTES	integer	Field length in bytes
SCALE	smallint	Scale for decimal field
LENFIELD	char(32)	Name of the field containing the length of this field
FLDEXIT	char(8)	Field exit name
FLDETYPE	char(2)	Field data type after field exit
FLDEBYTE	integer	Field length after field exit

Table 12. Columns of the FLD MVG Input Table and Their Data Types (continued)

Column	Data type	Description
FLDESCAL	smallint	Field scale after field exit
TABQUAL	char(8)	Qualifier of the propagated table
TABNAME	varchar(18)	Name of the propagated table
COLNAME	varchar(18)	Name of the propagated column

PRID (REQUIRED)
propagation request identifier. It must be the same value specified in the PRID column of the related propagation request row.

DBNAME (REQUIRED)
Identifies the database participating in propagation. It must be the same value specified in the DBNAME column of the related SEG table.

SEGNAME (REQUIRED)
Identifies the segment participating in propagation. It can be an IMS segment or an internal segment. It must be the same value specified in the SEGNAME column of the related SEG table.

FLDNAME (REQUIRED)
Identifies the field participating in propagation. Although this field can have any value, you should avoid using commas, semi-colons, and parentheses, as these can cause problems with other DPROP utilities.

POSITION (*n*) (REQUIRED)
Identifies the start position of the field in the segment. The first byte within the segment is considered to have the start position of one, not zero.

Replace *n* with a value indicating where the field begins in its segment. Unless the segment is processed by a Segment exit routine, use the following segment format to determine the starting position of the field: the segment format that IMS provides after it retrieves the segment with a DL/I call using a PCB that references the physical DBD without field sensitivity. The value of *n* can come from the START keyword of the DBDGEN FIELD statement defining the field.

For a logical child segment type, the segment always begins in the DL/I call I/O area with the fully concatenated key of the logical parent. This is true even if the fully concatenated key of the logical parent is defined in the DBD as virtual. Therefore, a field starting in position 1 starts in the fully concatenated key of the logical parent.

If you are describing a segment with a variable-length format, be sure to account for the 2-byte length field at the start of the segment. For example, a value of START=1 means it begins at the start of the length field, not at the byte following the length field.

Variable-length *internal* segments are not required to start with a 2-byte length field.

DATATYPE (OPTIONAL)
Use this column to specify the data type of the IMS field in IMS DPROP terminology. See Table 13 on page 113 for more information on data types.

DATATYP2 (OPTIONAL)
Use this column to specify the data type of the IMS field in DB2 terminology.

Table 13 on page 113 describes the data types supported by IMS DPROP in IMS DPROP and DB2 terminologies.

When you describe a field in the FLD table, you can specify its data type in IMS DPROP or DB2 terminology. You can also specify the data type in both IMS DPROP and DB2 terminology, but both terms must match, as shown in Table 13.

If you do not specify a value in the FLD table, the default values are DATATYPE=C and DATATYP2=CHAR.

Table 13. Data Types Supported by IMS DPROP

DATATYPE value of IMS fields (IMS DPROP terminology)	DATATYP2 value of IMS fields (DB2 terminology)	Type
A	DATE	date
B	B	single-byte binary data
C	CHAR	character
D	FLOAT	double precision floating point
E	FLOAT	single precision floating point
F	INTEGER	fullword
G	GRAPHIC	graphic
H	SMALLINT	halfword
P	DECIMAL	packed decimal
S	TIMESTMP	time stamp
T	TIME	time
VC	VARCHAR	variable-length character
VC	LONGVAR	variable-length character
VG	VARG	variable-length graphic
VG	LONGVARG	variable-length graphic
Z	Z	zoned decimal (unpacked)
xx	xx	User-defined data type. ‘xx’ must be two nonblank characters and must not be either VC or VG.

For fields having a 2-character data type, if the data type value is only one byte (for example, C), the byte should be left-justified (“C ” rather than “ C”).

Refer to the appropriate Administrators Guide for your propagation mode for details of data types and the data conversions between IMS fields and DB2 columns that are supported by IMS DPROP

BYTES (OPTIONAL or REQUIRED depending on the data type)
Identifies the length of the field, in bytes, or its maximum length if variable.

For fixed-length segments, IMS DPROP requires that propagated fields be totally contained in the segment; that is, the start and end of the field must be contained in the segment. For variable-length segments, each field must be either totally outside of the particular segment occurrence, or totally contained within the segment.

If you specify BYTES for fields whose lengths are determined by data type, you must specify the correct length as shown in Table 3 on page 42. If you specify BYTES for fields whose lengths are not determined by data type, you must specify a length in the ranges shown in Table 4 on page 42.

Note: VG and G data types represent double-byte character set (DBCS) data; therefore, you must use two bytes to represent one DBCS character when you calculate the length or starting position. When defining the length of a VG and G field for IMS DPROP, express the length in terms of bytes. However, when defining a graphic column to DB2, express its length in terms of the number of DBCS characters.

When defining the length of a packed decimal field for IMS DPROP, express the length in terms of bytes. However, when you define a decimal column to DB2, express its length in terms of digits.

Note: From IMS DPROP V1R2 and on, a length for user-defined, data type fields is also required in the BYTES column; the value is the length of the field in its user format.

SCALE (OPTIONAL)

Identifies the scale factor of the described field. Use this column for decimal fields only.

Indicates the position of the decimal point in the decimal representation of the field type.

The values you can specify are:

n If you want the decimal point to be placed to the left of the last *n* digits in the decimal representation. Replace *n* with a positive integer.

The value of *n* can range from 1 through the number of decimal digits contained in the field being described. For an *x*-byte field of type Z, the maximum scale, *n*, is *x*. For an *x*-byte field of type P, the maximum scale, *n*, is *2x-1*.

0 If there is no decimal point in the number (the value of the field is assumed to be an integer).

LENFIELD (REQUIRED for variable-length fields)

Name of the field containing the length of the variable-length field described by this row. LENFIELD must be specified if the field described by this FLD row is a variable-length field (if DATATYPE is VC or VG).

The field named here must precede the variable-length field within the IMS segment; that is, the value of the POSITION column of the FLD row describing the length field should be smaller than the POSITION column of the FLD row describing the variable length field.

The value of the field named here must be even if the field is defined as a VG data type.

The length field must be numeric data and must have a zero scale it if is packed or zoned decimal; it cannot be floating point. A length field can have a zero value.

FLDEXIT (OPTIONAL)

Identifies the name of a Field exit routine.

Leave FLDEXIT blank if the IMS field should not be processed by a Field exit routine. Otherwise, specify the name of the Field exit routine.

If you specify a Field exit routine, you must also specify the FLDETYPE. You can also specify FLDEBYTE and FLDESCAL.

IMS DPROP does not call Field exit routines for user mapping.

FLDETYPE (OPTIONAL OR REQUIRED depending on the data type)
Indicates the data type of the described field in its IMS DPROP format. Any value described in the first column of Table 13 on page 113 is valid, with the exception of xx.

FLDEBYTE (OPTIONAL)
Indicates the length in bytes of the described field in its IMS DPROP format.[4] For variable-length fields, indicate the maximum length.

You must specify a length that is valid for the data type identified in FLDEBYTE. Refer to Table 3 on page 42 and Table 4 on page 42 for valid field lengths.

FLDESCAL (OPTIONAL)
Indicates the scale factor of the described field in its IMS DPROP format.[5] FLDESCAL can be specified only for numeric fields in packed decimal or zoned decimal formats. See the description of the SCALE column on page 114 for information on valid scale factor specifications.

TABQUAL (OPTIONAL)
Identifies the propagated table qualifier. You must provide the same value here as the TABQUAL column in the related TAB row. Set to blanks if the TABQUAL column in the related TAB row contains blanks.

TABNAME (REQUIRED)
Identifies the propagated table name. You must provide the same value here as the TABNAME column in the related TAB row.

COLNAME (OPTIONAL)
Identifies the name of the column in the propagated table to or from which the described field is mapped. If you are describing a field which is not propagated, set COLNAME to blank.

Mapping Verification and Generation Utility (MVGU)

The Mapping Verification and Generation utility (MVGU) acts as a driver for mapping verification and generation (MVG) to create propagation requests (PRs) from the MVG input tables. The MVGU can also delete, rebuild and revalidate existing propagation requests.

Refer to Chapter 4, “Defining Propagation Requests Using MVG Input Tables ,” on page 91 for details on how to define a propagation request using the MVG input tables and the MVG input tables front end.

Overview

Use the Mapping Verification and Generation utility to:

- Create or replace one or several propagation requests (PRs) from the MVG input tables.
- Delete one or several propagation requests from the DPROP directory; the SQL update modules associated with the propagation requests are also deleted.
- Recreate propagation request control blocks (PRCBs) or SQL update modules from the mapping tables for one or several propagation requests.

4. For IMS-to-DB2 propagation, this is the format *after* processing by the Field exit routine. For DB2-to-IMS synchronous propagation, this is the format *before* processing by the Field exit routine.

5. For IMS-to-DB2 propagation, this is the format *after* processing by the Field exit routine. For DB2-to-IMS synchronous propagation, this is the format *before* processing by the Field exit routine.

- Revalidate one or several sets of logically-related propagation requests from the DPROP directory.

The Mapping Verification and Generation utility consists of a load module named EKYMVU00. It executes as a relational application running under the RUN CP subcommand of the DSN command.

Requirements

When the mapping verification and generation processes the following control statements, either the affected propagation requests in the DPROP directory must be inactive, or the IMS DPROP system must be emergency stopped:
- CREATE (resulting in the replacement of an existing propagation request)
- RECREATE
- DELETE

If affected propagation requests are active, call the SCU with the DEACTIVATE control statement before calling Mapping Verification and Generation utility.

Input and Output

Figure 43 illustrates the sources of input to the Mapping Verification Generation utility and the output created by this utility.

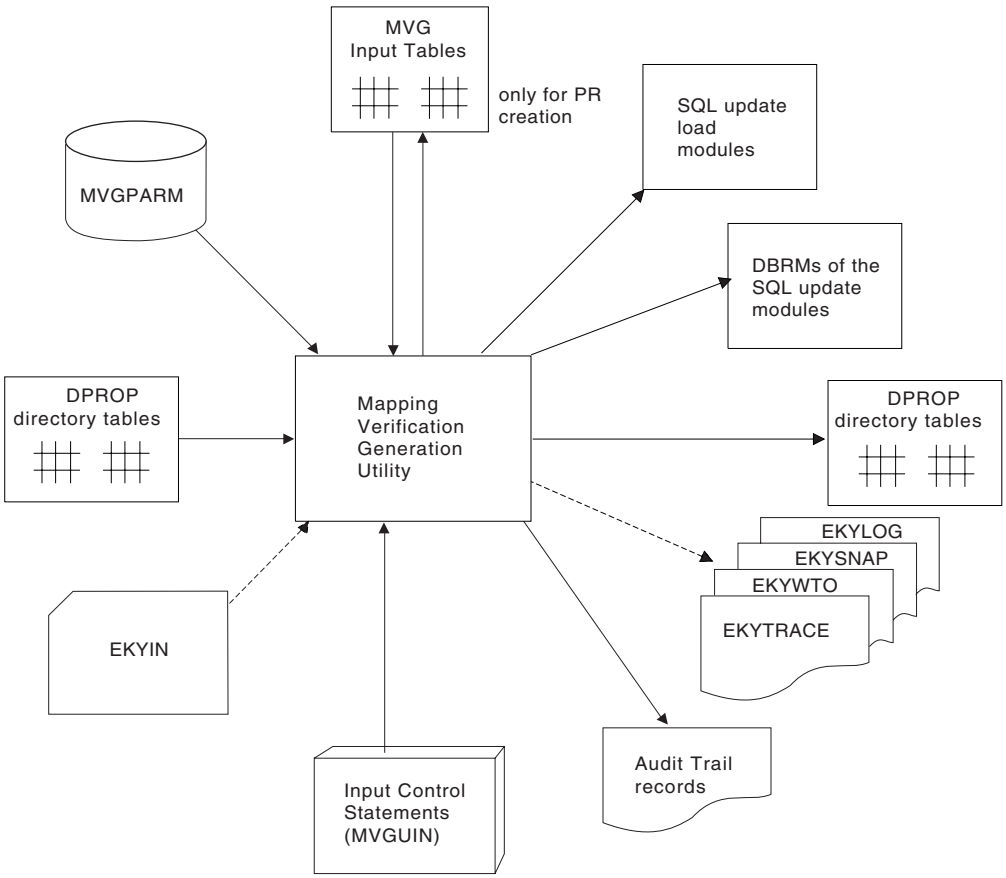

Figure 43. Mapping Verification Generation Utility

The input to the Mapping Verification and Generation utility consists of the following:

- The MVG input tables (PR, WHR, SEG, TAB, and FLD). The MVG input tables are used as input *only* for the creation or replacement of propagation requests.
- The IMS DPROP directory tables.
- The //MVGUIN data set containing the control statements (CREATE, RECREATE, DELETE or REVALIDATE).
- The //MVGPARM data set.
- The //EKYIN data set.

The output from the Mapping Verification and Generation utility consists of the following:

- The propagation request MVG input tables (only the PROCSED column).
- SQL update load modules.
- DBRMs of the SQL update modules.
- Packages of the SQL update modules.
- The IMS DPROP directory tables.
- The //EKYTRACE, //EKYWTO, //EKYSNAP and //EKYLOG files (the use of these files is optional, refer to IMS DPROP Diagnosis for details).
- Audit trail records (refer to IMS DPROP Messages and Codes for details).

Environment

The Mapping Verification and Generation utility executes as a relational application under MVS/ESA in 31-bit addressing mode (AMODE=31). It uses the TSO attach to connect to DB2. Mapping Verification and Generation utility must be called in the TSO environment by a RUN CP subcommand of the DB2 DSN command.

JCL Requirements

The JCL samples described in this topic are as follows:

- EKYUMVUP, the sample JCL procedure to execute the Mapping Verification and Generation utility
- EKYUMVUJ, the sample JCL job that calls EKYUMVUP

Sample JCL Procedure

Figure 44 on page 118 is the sample JCL procedure EKYUMVUP, which is used to execute the Mapping Verification and Generation utility.

```
//*-------------------------------------------------------------------*
//*       DPROP VERSION 2 RELEASE 1                                   *
//*  PROCEDURE EKYUMVUP TO EXECUTE THE MVGU                           *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*     ADAPT THE FOLLOWING VALUES AS REQUIRED                        *
//*     --------------------------------------                        *
//*                                                                   *
//*   1111111   - DPROP HLQ(S)                                        *
//*   2222222   - DBD LIBRARY                                         *
//*   3333333   - DBRM LIBRARY FOR SQL UPDATE MODULE                  *
//*   4444444   - LOAD LIBRARY FOR SQL UPDATE MODULE                  *
//*                                                                   *
//*   5555555   - MACLIB LIBRARY (IF MULTIPLE LIBRARIES ARE           *
//*               REQUIRED THEN THE PROCEDURE SHOULD BE AMENDED       *
//*               ACCORDINGLY).                                       *
//*   6666666   - LIB CONTAINING DPROPMIN MEMBER                      *
//*-------------------------------------------------------------------*
//*
//EKYUMVUP     PROC EKYPREF=1111111,
//             DBDLIB=2222222,
//             DBRMLIB=3333333,
//             LOADLIB=4444444,
//             MACLIB=5555555,
//             JCLLIB=6666666
```

Figure 44. Sample JCL Procedure EKYUMVUP (Part 1 of 2)

```
//*-------------------------------------------------------------------*
//MVGU         EXEC     PGM=IKJEFT01
//EKYRESLB     DD       DISP=SHR,
//             DSN=&EKYPREF..EKYRESLB
//*-------------------------------------------------------------------*
//* IN ASYNCH MODE, THERE IS NO STATUS FILE FOR THE DPROP SYSTEM      *
//* AND THE EKYSTATF DD STATEMENT CAN BE REMOVED IN THIS CASE.        *
//*                                                                   *
//EKYSTATF     DD       DISP=SHR,
//             DSN=&EKYPREF..STATFILE
//*-------------------------------------------------------------------*
//SYSTSPRT     DD       SYSOUT=*
//SYSTSIN      DD       DISP=SHR,
//             DSN=&JCLLIB(DPROPMIN)
//MVGPRINT     DD       SYSOUT=*
//DBDLIB       DD       DISP=SHR,
//             DSN=&DBDLIB
//MVGUMOD      DD       DSN=&&MVGUMOD,UNIT=SYSDA,
// DCB=(LRECL=80,BLKSIZE=6160,RECFM=FB),SPACE=(6160,(20,10),RLSE)
//PSYSPRT      DD       SYSOUT=*
//DBRMLIB      DD       DISP=SHR,
//             DSN=&DBRMLIB
//SYSTERM      DD       SYSOUT=*
//ASMIN        DD       DSN=&&ASMIN,UNIT=SYSDA,
// DCB=(LRECL=80,BLKSIZE=6160,RECFM=FB),SPACE=(6160,(10,5),RLSE)
//ASYSPRT      DD       SYSOUT=*
//*  ***  SEE ABOVE NOTES REGARDING THE FOLLOWING DD STATEMENT ***
//ASYSLIB      DD       DSN=&MACLIB,DISP=SHR
//ASYSLIN      DD       DSN=&&OBJECT,UNIT=SYSDA,
// DCB=(BLKSIZE=720,RECFM=FBS,BUFNO=30),SPACE=(720,(10,10),RLSE)
//LSYSPRT      DD       SYSOUT=*
//SYSLMOD      DD       DISP=SHR,
//             DSN=&LOADLIB
//SYSUT1       DD       UNIT=SYSDA,SPACE=(CYL,3)
//SYSUT2       DD       UNIT=SYSDA,SPACE=(CYL,3)
//SYSUT3       DD       UNIT=SYSDA,SPACE=(CYL,3)
```

Figure 44. Sample JCL Procedure EKYUMVUP (Part 2 of 2)

The DD statement //EKYRESLB, is common to many IMS DPROP components and are described in detail in Chapter 6, “IMS DPROP Directory Tables,” on page 139. The remaining DD statements are described below.

The PROC Statement

Defines the various keyword parameters of the JCL procedure. When calling EKYUMVUP, you must provide values for the keywords described below. Data set names must be enclosed in quotes.

EKYPREF The prefix for the data set names of the IMS DPROP libraries.

The EXEC Statement

Calls TSO batch to execute the Mapping Verification and Generation utility under DB2 using TSO Attach.

//EKYSTATF DD Statement

Is an optional statement that defines the IMS DPROP status file.

If you include this statement in your JCL job steps, Mapping Verification and Generation consults the status file to determine if the IMS DPROP system is in an emergency stopped state. If you do not provide this statement, Mapping Verification and Generation assumes that the IMS DPROP system is active.

If the system is in an emergency stopped state, //EKYSTATF for synchronous propagation allows you to:

- Replace propagation requests
- Process a RECREATE control statement
- Process a DELETE control statement

//SYSTSPRT DD Statement
Is a print file used by the TSO TMP environment.

//MVGPRINT DD Statement
Contains the messages issued by the Mapping Verification and Generation, MVGU, and MCE.

This data set is opened and closed each time a propagation request is processed. Therefore, when //MVGPRINT specifies a data set other than a sysout data set, it must be allocated with DISP=MOD.

//DBDLIB DD Statement
Specifies the DBD library that Mapping Verification and Generation accesses to validate and complete the database and segment information specified in the propagation requests to be processed.

//MVGUMOD DD Statement
Contains the source code of the SQL update modules that Mapping Verification and Generation creates. It is also used as input for the DB2 precompiler. //MVGUMOD replaces the usual //SYSIN statement of the precompiler.

//PSYSPRT DD Statement
Contains the print output of the DB2 precompile of the SQL update module. //PSYSPRT replaces the usual //SYSPRINT statement of the DB2 precompiler.

//DBRMLIB DD Statement
Describes the DBRM library where the DB2 precompiler stores the DBRM of the generated SQL update module. Do not provide the member name on the JCL statement, because Mapping Verification and Generation provides the name internally. The DBRM name provided by Mapping Verification and Generation is the same name as the propagation request ID. You will need to use this library when you bind the plans of the applications that perform propagation.

//SYSTERM DD Statement
Describes a print output file of the DB2 precompiler.

//ASMIN DD Statement
Contains the source code of the SQL update module after it has been processed by the DB2 precompiler. //ASMIN replaces the //SYSCIN statement of the precompiler and the //SYSIN statement of the assembler.

//ASYSPRT DD Statement
Contains the assembler listing of the SQL update module. //ASYSPRT replaces the usual //SYSPRINT statement of the assembler.

//ASYSLIB DD Statement
Describes the macro libraries required to assemble the SQL update module; that is, MACLIB for DB2, IMS DPROP, and MVS/ESA. //ASYSLIB replaces the //SYSLIB statement of the assembler.

//ASYSLIN DD Statement

Describes the object module of the SQL update module, which is passed to the linkage editor. //ASYSLIN replaces the //SYSLIN statement of the assembler.

//LSYSPRT DD Statement

Contains the print output of the linkage editor for the SQL update module. //LSYSPRT replaces the usual //SYSPRINT statement of the linkage editor.

//SYSLMOD DD Statement

Describes the load library where the linkage editor stores the load module of the SQL update module. Do not specify the member name of the load module in your JCL, because Mapping Verification and Generation provides it internally. The load module name provided by Mapping Verification and Generation is the same name as the propagation request ID. The job steps performing synchronous or asynchronous propagation must be able to access the load module stored in this library.

//SYSUT1, //SYSUT2, //SYSUT3 DD Statements

These three statements are work files used by the DB2 precompiler, the assembler, and the linkage editor.

Sample JCL To Call the EKYUMVUP JCL Procedure

Figure 45 on page 122 shows sample JCL statements that call the EKYUMVUP JCL procedure to execute the Mapping Verification and Generation utility.

```
//JJJJJJJ  JOB *** VALID JOB CARD DETAILS ***
//*-------------------------------------------------------------------*
//*           DPROP VERSION 2 RELEASE 1                               *
//*   JOB EKYUMVUJ TO EXECUTE THE MVGU PROCEDURE (EKYUMVUP)           *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S NEED:          *
//*  -------------------------------------------------------          *
//*                                                                   *
//*  JJJJJJJ   - JOB NAME                                             *
//*  1111111   - MVU CONTROL STATEMENTS AS REQUIRED                   *
//*  ADAPT THE MVGPARM CONTROL STATEMENT AS REQUIRED                  *
//*                                                                   *
//*-------------------------------------------------------------------*
//EKYUMVU      EXEC EKYUMVUP
//*-------------------------------------------------------------------*
//MVGUIN   DD   *
1111111
/*
//MVGPARM   DD  *
PRTYPE=L,MAPCASE=1,MAPDIR=HR,ERROPT=BACKOUT ;
/*
```

Figure 45. Sample JCL Job, EKYUMVUJ, to Run the Mapping Verification and Generation utility

//SYSTSIN DD Statement

Is the input data set for TSO batch. It contains a DSN command, a RUN CP subcommand, the Mapping Verification and Generation utility program name that TSO processes as a TSO command, and an END command.

This data set is primarily used to execute the Mapping Verification and Generation utility as a DB2 application under TSO batch.

If you have more than one DB2 subsystems installed, IMS DPROP also uses this data set to get the name of the correct DB2 subsystem to perform the BIND PACKAGE or FREE PACKAGE of the SQL update module.

Figure 46 on page 123 shows a sample member containing the TSO input statements used to execute the MVGU.

Note: To be read by IMS DPROP, this data set must specify an MVS sequential file or a member of a partitioned data set (for example, the member DPROPMIN), not an instream data set.

If //SYSTSIN specifies an instream data set, IMS DPROP uses the name of the default DB2 subsystem, generally DSN, which is specified at DB2 installation time.

//MVGUIN DD Statement

Contains the Mapping Verification and Generation utility input control

statements that specify propagation requests to create, delete, recreate or revalidate. This data set can contain several control statements. For information on the control statements, refer to “Control statements ” on page 123.

//MVGPARM DD Statement

Contains propagation parameter defaults. Mapping Verification and Generation accesses it to find default values for the propagation parameters. Mapping Verification and Generation will use the defaults in this data set if you do not provide all the propagation parameters on the propagation request mapping verification and generation input tables.

Refer to Chapter 5, “//MVGPARM Data Set Propagation Parameters,” on page 133 for more information on the //MVGPARM data set.

Sample DPROPMIN Member To Execute the MVGU

Figure 46 shows the sample DPROPMIN member. It contains control statements required to execute the MVGU as a DB2 application under the TSO attach with the RUN CP subcommand. This member is allocated by the //SYSTSIN DD statement in the EKYUMVUP JCL procedure.

```
DSN SYSTEM( dsn)
 RUN CP PLAN(plan-name)
  EKYMVU00
 END
```

Figure 46. Sample DPROPMIN member to execute the MVGU

You can adapt this sample to your installation needs as follows:

- Replace *dsn* with the name of your DB2 subsystem.
- The CP option is mandatory under the RUN subcommand and must be coded as shown.
- Replace *plan-name* with the name of the DB2 plan used to execute the MVGU.
- EKYMVU00 is the program name of the MVGU and must be coded as shown.

If more than one DB2 subsystem is installed, the data set specified with //SYSTSIN DD statement should be an MVS sequential file or a member of a partitioned data set (for example, the member DPROPMIN). This is so IMS DPROP can get the name of the correct DB2 subsystem to perform the BIND PACKAGE or the FREE PACKAGE function.

Control statements

Table 14 on page 124 summarizes the processing performed for the different types of Mapping Verification and Generation utility control statements.

Table 14. Summary of Processing of Mapping Verification and Generation utility control statements

Mapping Verification and Generation utility Control Statement	Tables used as Input	PR in Mapping Verification and Generation Input Table	PR in Mapping Tables	PRCB in CB tables and in VLF	SQL Update Load Module & DBRM[1]	Package of SQL Update Module[2]
CREATE	mapping verification and generation input tables	Flagged as processed	Added /Replaced	Added /Replaced	Added /Replaced	Added /Replaced
RECREATE	Mapping tables		PR row replaced, if default BIND options used	Replaced	Replaced	Replaced
DELETE	Mapping tables	-	Deleted	Deleted	Deleted	Deleted
REVALIDATE	Mapping tables	-	PR row replaced, MSG rows added /replaced	-	-	-

[1]Only for a propagation request with MAPDIR=HR or TW belonging to a generalized mapping case.
[2]Only for a propagation request.with MAPDIR=HR or TW belonging to a generalized mapping case *and* only if package bind option is selected.

CREATE

To create one or more propagation requests in the IMS DPROP directory, use the CREATE statement of the Mapping Verification and Generation utilityIN data set. The propagation requests and the associated SQL update modules are created based on the content of the mapping verification and generation input tables. This control statement can be used only for those propagation requests defined with the mapping verification and generation input tables.

Mapping Verification and Generation utility only creates propagation requests that have a value other than Y in the PROCSED column of the propagation request mapping verification and generation input tables; that is, Propagation requests that are not yet processed by the Mapping Verification and Generation. After successful creation of the propagation request, Mapping Verification and Generation utility sets the PROCSED column of the PR mapping verification and generation input tables to Y.

At a later time, you might want to replace the propagation request. You must first update the PROCSED column by setting it to a value other than Y, and then update the content of the mapping verification and generation input tables based on desired values.

If the propagation requests in question have to be replaced (that is, if they already exist in the IMS DPROP directory), either all these propagation requests must be inactive or the whole DPROP system must be emergency stopped.

The syntax for the CREATE statement is shown in Figure 47 on page 125.

```
                    ┌──,──┐
                    ▼     │
►►─CREATE──┬─PR=──(──pr_id──)──────────────;──────────────────────►◄
           │
           │                ┌──,──┐
           ├─DBD=dbd,SEG=──(──segment──)──┤
           │
           │            ┌──,──┐
           ├─DBD=──(──dbd──)──────────────┤
           │                              │
           └─ALL──────────────────────────┘
```

Figure 47. The CREATE control statement

PR=(*prid1,prid2,...* **)**
Identifiers for propagation requests to be created

If the PROCSED column of the propagation request mapping verification and generation input tables has a value other than Y, the propagation requests identified here are created/replaced in the order in which they appear on the PR= keyword.

DBD= *database* **,SEG=(***segname1,segname2,...***)**
If you want to create all the propagation requests not yet processed, but already coded in the mapping verification and generation input tables that propagate a specified segment, identify here the name of the segment and the database to which it belongs.

The segments specified with SEG= can only be IMS segments, not internal segments. If you want to create a propagation request for a specific internal segment, you have to specify it on the PR= keyword.

If the specified segment is a containing IMS segment (a segment that contains propagated internal segments), then all propagation requests not yet processed that propagate the dependent internal segments are also created/replaced.

If an IMS segment identified in the SEG= keyword contains fields used as path data, then:

- The control statement is applied *only* to those Propagation requests propagating the identified segment as an entity or extension segment (and those propagation requests propagating internal segments contained in the identified IMS segment).
- The control statement is not applied to those propagation requests propagating dependent segments and including path data from the identified segment.

The identified segments are processed in the order in which they appear on the SEG= keyword.

DBD=(*database1,database2,...* **)**
Creates all the propagation requests not yet processed, but already coded in the mapping verification and generation input tables that propagate specified databases.

For each database, the relevant propagation requests are processed from the mapping verification and generation input tables in the following order:

- Level by level, from the root to the bottom of the database.
- Within each level the logical parents, if any, are processed first.

- Next, the other segments on the same level are processed in the sequence of the segment code (left to right).

And so on, until all propagation requests propagating the specified DBD have been processed.

ALL

Creates all the propagation requests not yet processed but already coded on the mapping verification and generation input tables.

The order of processing is as follows:

- Level by level, over all of the databases found, from the root of the first database found to the bottom of the last database found.
- Within each level the logical parents, if any, are processed first.
- Next, the other segments in the same level over all of the databases are processed in the sequence of the segment code (left to right).

And so on, until all propagation requests are processed.

DELETE

Use the DELETE statement of the Mapping Verification and Generation utility data set to delete

- one or more propagation requests from the DPROP directory
- the SQL update modules from the SYSLMOD and DBRMLIB libraries of the Mapping Verification and Generation utilityIN data set.

The Mapping Verification and Generation utility does *not* delete a propagation request from the mapping verification and generation input tables.

Note: For LOG-ASYNC, you cannot delete a propagation request while it is assigned to a Receiver in the propagation request control table. This is because of the DB2 Referential Integrity rule that applies to the propagation request control table and the propagation request table. Refer to Chapter 8, “Status Change Utility (SCU) for LOG-ASYNC ,” on page 213 for details on deleting the row that corresponds to the propagation request from the propagation request control table.

You can only use the DELETE statement if all concerned propagation requests are inactive or the DPROP system is emergency stopped.

You can also use this control statement if you defined your propagation request with DataRefresher.

The syntax for the DELETE statement is shown in Figure 48 on page 127.

```
►►─DELETE──┬─PR=──(──┬──pr_id──┬──)──────────────────────────┬──────────────────────►
           │         └─────────┘ │                           │ └─,PRSET=prset──┘
           │                     │                           │
           ├─DBD=dbd,SEG=──(──┬──segment──┬──)──────────────┤
           │                  └───────────┘                 │
           │                                                │
           └─DBD=──(──┬──dbd──┬──)──────────────────────────┘
                      └───────┘

►──┬──────────────────────────────────────────────────────┬──;──────────────────────►◄
   │                                                      │
   └─PACKAGE=─(──┬──collection-id──┬──)──────────────────┘
                 └─────────────────┘
```

Figure 48. The DELETE control statement

PR=(*prid1,prid2,...* **)**
Identifier for propagation requests to be deleted.

DBD= *database* **,SEG=(***segname1,segname2,...***)**
Deletes all the propagation requests existing on the DPROP directory and propagating a specific segment.

The segments specified with SEG= can only be IMS segments, not internal segments. If you want to delete a propagation request for a specific internal segment, you have to specify it on the PR= keyword.

If the specified segment is a containing parent segment (a segment that contains propagated internal segments), then all propagation requests propagating the dependent internal segments are also deleted, even if the containing segment is not propagated.

If an IMS segment specified in the SEG= keyword contains fields used as path data, then:

- The control statement is applied *only* to those propagation requests propagating the identified segment as an entity or extension segment (and those propagation requests propagating internal segments contained in the specified IMS segment).
- The control statement is not applied to those propagation requests propagating dependent segments and including path data from the specified segment.

DBD=(*database1,database2,...* **)**
Deletes all the propagation requests existing on the DPROP directory and propagating a specific database.

PRSET=prset
Used with PR=, DBD=, or SEG= to limit the deletion of propagation requests belonging to a specific set of propagation requests (PR set).

PACKAGE=(*collection-id1,collection-id2,...***)**
Tells Mapping Verification and Generation from which collection IDs it must remove the packages of the SQL update module. Mapping Verification and Generation uses this parameter to dynamically execute FREE PACKAGE commands.

If you do not specify a collection ID, no package is deleted.

RECREATE

To recreate the PRCBs in the control block tables of the IMS DPROP directory, or the elements of the SQL update modules (load module, DBRM, bind package) or both, use the RECREATE statement of the Mapping Verification and Generation utilityIN data set. PRCBs and SQL update modules are recreated based on the content of the mapping tables of the IMS DPROP directory, *not* on the content of the mapping verification and generation input tables or the DataRefresher definitions. For information on the DPROP directory, refer to refid=dprjcl..

The RECREATE function is typically used only when a propagation control block has been destroyed or an element of the SQL update module (load module, DBRM, bind package) has been deleted by mistake.

The RECREATE function can only be performed if all concerned propagation requests are inactive or the DPROP system is emergency stopped.

You can also use this control statement if you defined your propagation requests with DataRefresher.

The syntax for the RECREATE statement is shown in Figure 49.

```
                          ┌──,──┐
                          ▼     │
►►─RECREATE──┬─PR=──(──────pr_id──────)──────────────────────────────────►
             │                                                  ┌─,CBTABLE─┐
             │                          ┌──,──┐                 └─,SQLMOD──┘
             ├─DBD=dbd,SEG=──(──────segment──────)──┤
             │                          ┌──,──┐
             ├─DBD=──(──────dbd──────)──────────────┤
             └─ALL──────────────────────────────────┘

►─┬──────────────────┬──;──────────────────────────────────────────────►◄
  └─,BIND=DEFAULT────┘
```

Figure 49. The RECREATE control statement

PR=(*prid1,prid2,...* **)**
Propagation request identifier to be recreated

DBD= *database name* **,SEG=(***segname1,segname2,...***)**
Recreates the PRCBs or SQL update modules or both for the propagation requests propagating a specific segment.

The segments specified with SEG= can only be IMS segments, not internal segments. If you want to recreate a CBTABLE or an SQL update module or both for a specific internal segment, you have to specify it on the PR= keyword.

If the specified segment is a containing parent segment (a segment that contains propagated internal segments), then all propagation requests propagating the dependent internal segments are also recreated, even if the containing segment is not propagated.

If an IMS segment specified in the SEG= keyword contains fields used as path data, then:

- The control statement is applied *only* to those propagation requests propagating the identified segment as an entity or extension segment (and those propagation requests propagating internal segments contained in the specified IMS segment).
- The control statement is not applied to those propagation requests propagating dependent segments and including path data from the specified segment.

DBD=(*database1,database2,...* **)**
Recreates the PRCBs or SQL update modules or both for the propagation requests existing on the DPROP directory and propagating a specific database.

ALL
Recreates all the PRCBs or SQL update modules or both for the propagation requests existing in the mapping tables.

CBTABLE | SQLMOD
To recreate only control block table rows, select CBTABLE. To recreate only the SQL update module, select SQLMOD. If you want to recreate both the control block table rows and the SQL update module, *do not* select either keyword.

BIND=DEFAULT
This keyword is relevant only for propagation requests that do not have BIND options in the propagation request mapping table row.

For such propagation requests, DPROP reads the //Mapping Verification and GenerationPARM data set, which contain s default propagation parameters, if that data set is allocated. IMS DPROP obtains BIND options from this data set, if any are specified.

If //Mapping Verification and GenerationPARM is allocated and contains BIND options (at least the collection ID), a PACKAGE BIND is performed for the current propagation request, and the bind options are stored in the propagation request mapping table.

If //Mapping Verification and GenerationPARM is not allocated or does not contain BIND options, no PACKAGE BIND is performed, but the RECREATE function is completed.

Package Bind

If the SQL update module is to be recreated, and bind options were specified at propagation request creation time, then the bind package of the SQL update module is also recreated.

The bind options specified at propagation request creation time are stored in the propagation request mapping table. To recreate the bind package of the SQL update module, Mapping Verification and Generation uses these bind options to dynamically execute the BIND PACKAGE command.

REVALIDATE

Use the REVALIDATE statement to check that the referential integrity rules of the DB2 propagated tables are consistent with the hierarchy of the propagated IMS/DBs.

This function is also used to revalidate propagation requests that were defined earlier. The accuracy of some propagation request definitions could be in question if the definition of a DB2 propagated table or an IMS DBD has been modified.

The revalidation is based on the content of the mapping tables of the IMS DPROP directory, not on the content of the mapping verification and generation input tables or the DataRefresher definitions. For information on the DPROP directory, refer to Chapter 6, “IMS DPROP Directory Tables,” on page 139.

Propagation requests to be revalidated do not need to be inactive.

You can also use the REVALIDATE function if you have defined your propagation requests with DataRefresher.

The syntax for the REVALIDATE statement is shown in Figure 50.

```
                        ┌─,──────┐
                        ▼        │
►►─REVALIDATE──┬─PR=──(──pr_id──────)──────────────┬──────────────────────┬──;──────►◄
               │                                   │                      └─,BIND=DEFAULT─┘
               │                  ┌─,──────┐       │
               ├─DBD=dbd,SEG=──(──▼segment────)────┤
               │                  ▼        │       │
               │                  ┌─,──────┐       │
               ├─DBD=──(──dbd──────)──────────────┤
               └─ALL──────────────────────────────┘
```

Figure 50. The REVALIDATE control statement

PR=(*prid1,prid2,...* **)**
Identifiers for propagation requests to be revalidated.

The identified propagation requests are revalidated in the order in which they appear on the PR= keyword.

DBD= *database* **,SEG=(***segname1,segname2,...***)**
If you want to revalidate all the propagation requests on the IMS DPROP directory that propagate a specified segment, specify here the name of the segment and the database to which it belongs.

The segments specified with SEG= can only be IMS segments, not internal segments. If you want to revalidate a propagation request for a specific internal segment, you must specify it on the PR= keyword.

If the specified segment is a containing IMS segment (a segment that contains propagated internal segments), then all propagation requests on the DPROP directory that propagate the dependent internal segments are also revalidated.

If an IMS segment specified in the SEG= keyword contains fields used as path data, then:

- The control statement is applied *only* to those propagation requests propagating the identified segment as an entity or extension segment (and those propagation requests propagating internal segments contained in the specified IMS segment).
- The control statement is not applied to those propagation requests propagating dependent segments and including path data from the specified segment.

The identified segments are processed in the order in which they appear on the SEG= keyword.

DBD=(*database1,database2,...* **)**

Revalidates the propagation requests propagating a specific database.

Unlike the CREATE statement, for each database, the relevant propagation requests are processed from the mapping tables in the order of the DB hierarchy as follows:

- First, the propagation requests propagating the root segment,
- Next, the propagation requests propagating the first immediate dependent,
- And so on, in the sequence of the segment code, until all propagation requests propagating the specified DBD have been processed.

ALL

Revalidates all the propagation requests existing on the IMS DPROP directory.

Unlike the CREATE statement, all propagation requests are processed from the mapping tables, DBD by DBD, in the order of the DB hierarchy.

Notes on Using REVALIDATE: The REVALIDATE function does not affect the status of the propagation request on the IMS DPROP directory, that is, the propagation request is not stopped if the revalidation finds any errors.

Mapping Verification and Generation utility writes errors on the //MVGPRINT data set and on the MSG mapping table, and also creates audit records.

The only IMS DPROP directory tables affected by the REVALIDATE function are the propagation request and the MSG table.

The PR table

After a propagation request is revalidated, information is provided in the following columns of the propagation request table for the propagation request in question:

- REVALRES gives the result of the revalidation.

  I Neither errors nor warnings were found during the revalidation.

  W The most serious message issued was a warning.

  E At least one error was found during the revalidation.

  blank This propagation request has never been revalidated.
- REVALTMS is the timestamp of the revalidation. This column is only meaningful if the propagation request has been revalidated, that is, if REVALRES is not blank. If the propagation request has been revalidated a second, a third or *nth* tim e, the new revalidation timestamp replaces the old one.

The MSG table

The MSG table can contain messages issued during the creation of the propagation request and during the revalidation of the propagation request.

After a revalidation:

- Warning messages, if any, issued during the propagation request creation remain in the MSG table.
- Warning or error messages, if any, issued during the PR revalidation are also stored in the MSG table.

To distinguish between these two kind of messages, see the ORIGIN column. It contains information about the origin of the messages inserted in the MSG table, as follows:

- A blank means that the message was issued during the propagation request creation.
- An R means that the message was issued during the propagation request revalidation.

If a propagation request is revalidated a second, third or *nth* time, then the messages provided in the previous revalidation are deleted, and the new messages, if any, are inserted.

Output Messages and Statistics

The error messages issued by Mapping Verification and Generation utility begin with EKYV. For a listing of the error messages issued by Mapping Verification and Generation utility, refer to IMS DPROP Messages and Codes.

Return Codes and Error Conditions

The Mapping Verification and Generation utility provides the following return codes:

Code	Meaning
0	All Mapping Verification and Generation utility control statements successfully executed.
8	One of the following occurred: • No Mapping Verification and Generation utility control statement was specified. • The Mapping Verification and Generation utility control statement was invalid. • Some propagation requests were processed, but at least one was not processed successfully.
16	A severe error occurred. The Mapping Verification and Generation utility terminates.

A return code of 0 does not imply error-free execution of the Mapping Verification and Generation utility. Mapping Verification and Generation can have issued warning messages along with the Mapping Verification and Generation utility return code of 0. Therefore, you should check your job output or query the message table for more information on possible warning messages.

Chapter 5. //MVGPARM Data Set Propagation Parameters

To create a valid propagation request, specify the parameters described in this topic.

If you code your propagation request via DataRefresher, provide the Propagation parameters on the MAPUPARM keyword of the DataRefresher SUBMIT command. If you code it on the MVG input tables, specify the Propagation parameters in the propagation request MVG input table.

Whether you use DataRefresher or the MVG input tables to code your propagation request, you can omit one, several, or all of the Propagation parameters at MVG execution time. MVG accesses the //MVGPARM data set, which contains propagation parameter default values.

The following topics provide additional information:
- “Required Propagation Parameters ” on page 134
- “Default Values for Propagation Parameters ” on page 135

Figure 51 on page 134 shows the propagation parameters provided on the //MVGPARM data set. Code these Propagation parameters the same as you code the parameters on the MAPUPARM keyword of the DataRefresher SUBMIT command.

For a description of these Propagation parameters, see the MAPUPARM keyword in Chapter 3, “Defining Propagation Requests Using DataRefresher ,” on page 25.

```
   ┌──,──────────────────────────────────────┐
   ▼                                          │
►►─┬─PRTYPE=──┬─E─┬──────────────────────────┬─;──────────────────────────────►◄
   │          ├─L─┤                          │
   │          ├─F─┤                          │
   │          └─U─┘                          │
   ├─MAPCASE=──┬─1─┬──────────────────────────┤
   │           ├─2─┤                          │
   │           └─3─┘                          │
   ├─PATH=──┬─ID─────────────────────────────┤
   │        └─DENORM──────────────────────────┘
   ├─MAPDIR=──┬─HR─┬──────────────────────────┤
   │          ├─TW─┤                          │
   │          └─RH─┘                          │
   ├─TABQUAL2=qualifier──────────────────────┤
   ├─ERROPT=──┬─BACKOUT─┬──────────────────────┤
   │          └─IGNORE──┘                     │
   ├─MAXERROR=number─────────────────────────┤
   ├─ACTION=──┬─ADD──┬──────────────────────┤
   │          └─REPL─┘                        │
   ├─PERFORM=──┬─BUILDRUN──┬──────────────────┤
   │           ├─BUILDONLY─┤                  │
   │           └─RUNONLY───┘                  │
   ├─PRSET=setname───────────────────────────┤
   ├─PROPSUP=──┬─N─┬──────────────────────────┤
   │           └─Y─┘                          │
   ├─EXITNAME=exitname───────────────────────┤
   ├─PROPSEGM=(dbd/seg,...)──────────────────┤
   ├─KEYORDER=──┬─ASC─┬──────────────────────┤
   │            ├─DSC─┤                      │
   │            └─ANY─┘                      │
   ├─PCBLABEL=pcblabel───────────────────────┤
   ├─BIND=(bind options)─────────────────────┤
   ├─AVU=──┬─Y─┬──────────────────────────┤
   │       └─N─┘                              │
   ├─DEFVEXT=──┬─Y─┬──────────────────────────┤
   │           └─N─┘                          │
   └─COMMENT=(comments)────────────────────────┘
```

Figure 51. //MVGPARM Data Set

Required Propagation Parameters

Some parameters must always be provided in the //MVGPARM data set, on the DataRefresher SUBMIT command, or in the MVG propagation request input table. These are:

- PRTYPE (E|L|F|U)
- MAPCASE (1|2|3)
- MAPDIR (HR|TW|RH)
- ERROPT (BACKOUT|IGNORE)

If you do not specify these parameters on the DataRefresher SUBMIT command or in the MVG propagation request input table, MVG must find them in the //MVGPARM data set. Otherwise, MVG issues an error message.

If you use your own mapping (PRTYPE=U), the MAPCASE parameter is ignored but must either contain a valid value or be set to blank.

Default Values for Propagation Parameters

IMS DPROP assigns default values to the remaining propagation parameters, listed in Table 15. IMS DPROP assigns default values only if you do not specify values on the DataRefresher SUBMIT command, on the MVG input tables, and in the //MVGPARM data set.

The default value of some parameters, however, is not valid for all types of propagation requests. If an assigned default value does not conform to the rules of the propagation requests you are creating, IMS DPROP issues an error message and the propagation request is not created. To correct this problem, specify the parameter value on the DataRefresher SUBMIT command, on the MVG input tables, or in the //MVGPARM data set.

Assigned default values are not valid for propagation requests that propagate PATH data with generalized mapping logic, such as the default value of the PATH= propagation parameter.

Table 15. Default Values for the Remaining Propagation Parameters

Parameter	Default Value
PATH	Blank; path-data is *not* allowed in the propagation request.
TABQUAL2	The userid of the user running the job
MAXERROR	No limitation of error messages
ACTION	Add; if the propagation request already exists, MVG does not replace it, but issues an error message.
PERFORM	BUILDRUN; for DataRefresher users only
PRSET	The propagation request set specified at IMS DPROP customization
PROPSEGM	No segment is propagated; at least one segment is required for a user mapping
EXITNAME	No exit name; an exit is required for user mapping
KEYORDER	ASC
PCBLABEL	If the propagation request is specified via DataRefresher, it is the name of the DXTPCB that is used as default. If the propagation request is coded on the MVG input tables, there is no default value.
BIND	No bind option
AVU	Blank
DEFVEXT	Y or blank
COMMENT	Blank

Part 3. IMS DPROP Directory Tables

Chapter 6. IMS DPROP Directory Tables

Notice
This topic contains general-use programming interface and associated guidance information.

This topic provides details about how the IMS DPROP directory tables are updated, what the tables contain, where the tables exist, when they were created, and examples of how to query tables for information.

The IMS DPROP directory is a set of relational tables that store mapping information about propagation requests. The IMS DPROP directory tables are used by:

SCU To process control statements that assign propagation requests to particular receivers.

RUP To obtain mapping information

The Receiver To control grouping of updates, and for recovery

You supply mapping information about propagation requests to the Mapping Verification and Generation utility (MVGU) via DataRefresher or the MVG input tables. MVGU validates the mapping definitions and stores them in the IMS DPROP directory control block tables and in the mapping tables.

The following topics provide additional information:

- “Updating IMS DPROP Directory Tables ” on page 142
- “The Master Table (DPRMASTER) ” on page 143
- “The RUP Control Block Table (DPRRCBT) ” on page 143
- “The HUP Control Block Table (DPRHCBT) ” on page 144
- “Mapping Tables ” on page 145
- “Control Tables Used by LOG-ASYNC ” on page 161
- “Examples of How to Query IMS DPROP Tables ” on page 165

If you have more than one IMS DPROP system you must create IMS DPROP directory tables for each system. Each IMS DPROP system is a separate entity with its own IMS DPROP directory tables.

The tables in the IMS DPROP directory are:
- Master table
- RUP control block table (CBT)
- HUP CBT
- Mapping tables:
  - Propagation request mapping table
  - MSG mapping table
  - WHR mapping table
  - SEG mapping table
  - TAB mapping table
  - FLD mapping table
- Tables used by LOG-ASYNC:
  - Receiver control table (RCT)
  - Propagation request control table (PRCT)
  - PRDS register table
  - PRDS volume table

The relationship between the tables is illustrated in Figure 52 on page 141 and Figure 53 on page 142.

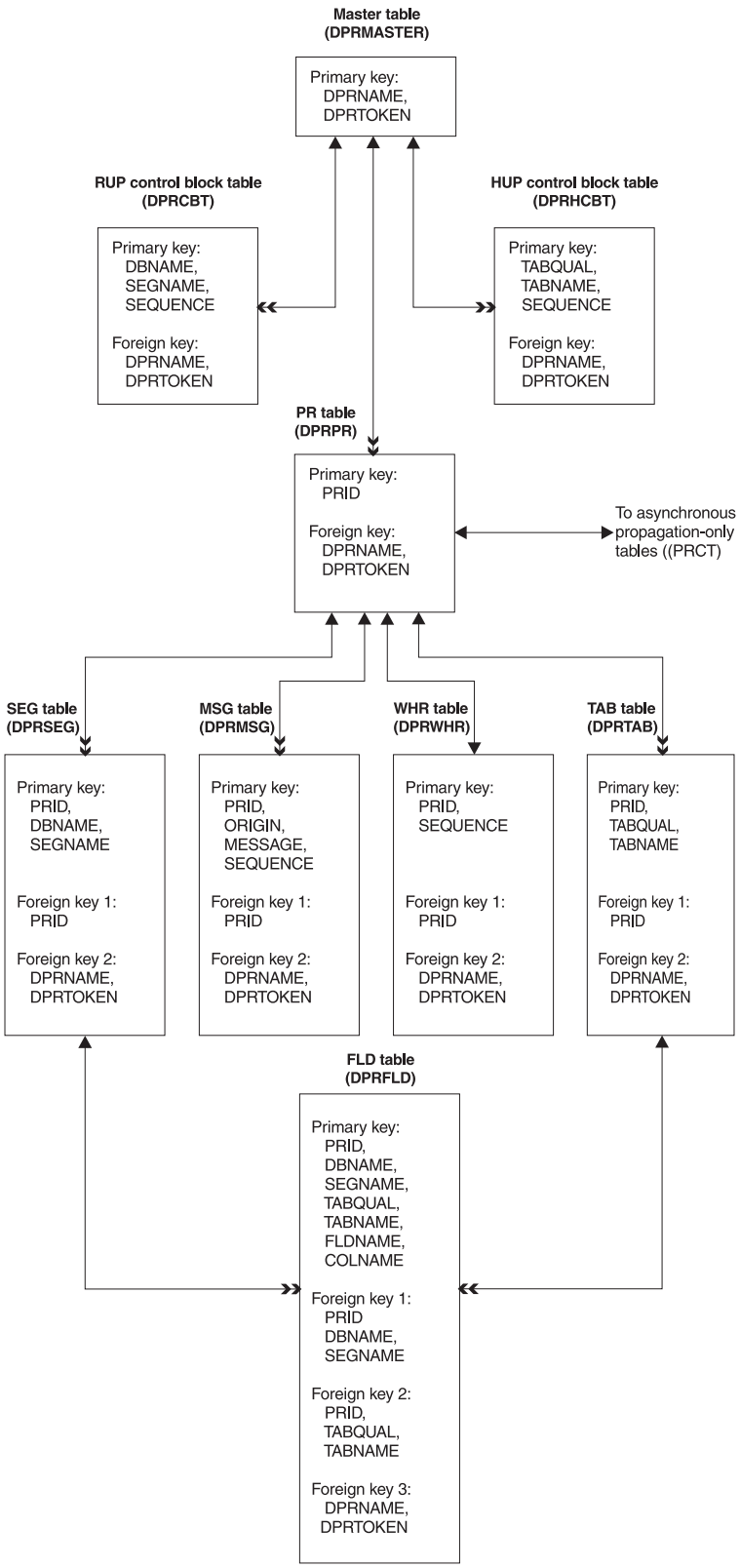

Figure 52. IMS DPROP Directory: Common Tables

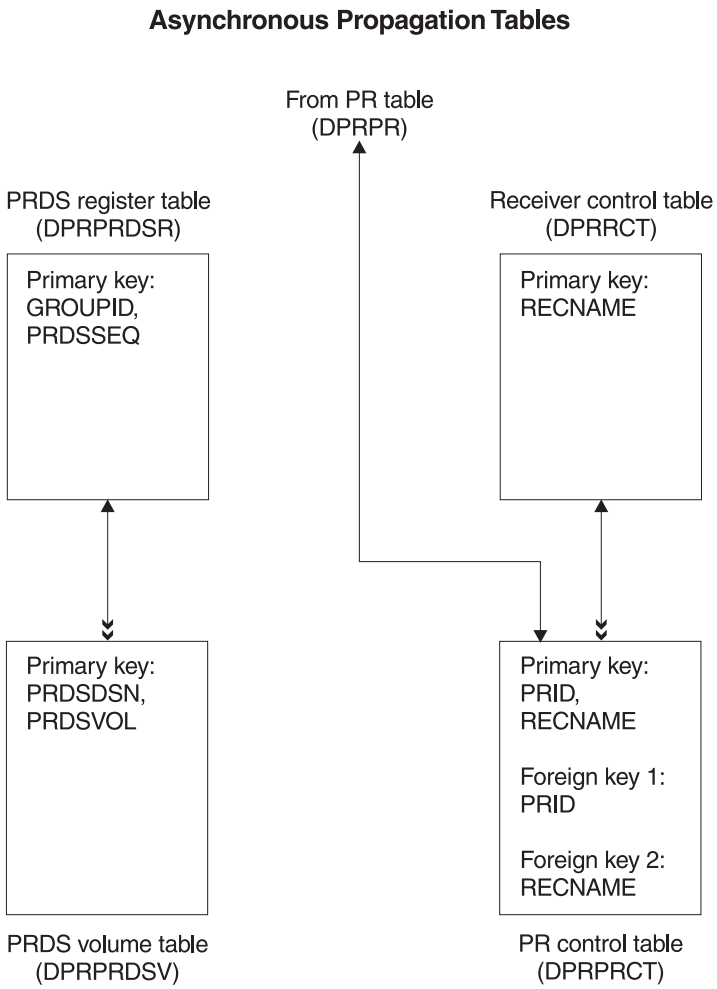

Figure 53. IMS DPROP Directory: Asynchronous Propagation Tables

Updating IMS DPROP Directory Tables

The IMS DPROP directory tables are updated by different IMS DPROP components:

- The PRU updates the PRDSREG exclusively.
- The Receiver and the SCU update the RCT.
- The MVG and SCU update all other tables exclusively.

Do not update the IMS DPROP directory tables with your own applications or query management facility (QMF™). If you do, propagation failures can occur and there can be inconsistencies in the data in:

- The IMS DPROP directory tables
- The IMS DPROP directory tables versus the VLF
- The IMS DPROP directory tables versus SQL update modules

Instead of updating the tables with your own applications or QMF, you should use IMS DPROP utilities such as the MVG or the SCU.

The Master Table (DPRMASTER)

The MASTER table is created when an INIT IMS DPROP control statement calls the SCU. It only contains one row, and has five columns. Table 16 describes the columns in the MASTER table.

The DPRNAME and DPRTOKEN columns compose the primary key of the MASTER table. These columns are referred to as a foreign key in the RUP CBT, propagation request, MSG, WHR, SEG, TAB, and FLD mapping tables.

Table 16. Columns of the Master Table and Their Data Types

Column	Data Type	Description
DPRNAME	char(8)	IMS DPROP system name
DPRTOKEN	dec(15,0)	IMS DPROP system token
TIMESTM	timestamp char(26)	IMS DPROP master timestamp
MAXWTO	integer	Number of errors on console
MAXAUD	integer	Number of errors on audit

The columns in DPRMASTER are:

DPRNAME
The name of the IMS DPROP system.

DPRTOKEN
A unique IMS DPROP-assigned token associated with the DPRNAME.

TIMESTM
The master timestamp for the IMS DPROP system. It is updated with each change to the IMS DPROP directory.

MAXWTO
The maximum number of errors for propagation requests in which ERROPT=IGNORE that are reported to the OS/VS console in a 15 minute interval for each IMS DPROP system.

MAXAUD
The maximum number of errors for propagation requests in which ERROPT=IGNORE that are reported to the IMS DPROP audit trail in a 15 minute period for each IMS DPROP system.

The RUP Control Block Table (DPRRCBT)

There is one RUP CBT for each IMS DPROP directory.

The RUP CBT contains one RUP propagation control block (RUP PRCB) for each propagated segment type. IMS DPROP generates, deletes, or regenerates rows in the RUP CBT when it creates, deletes, or recreates propagation requests.

The DBNAME, SEGNAME, and SEQUENCE columns compose the primary key of the RUP CBT. The DPRNAME and DPRTOKEN columns are referred to as a foreign key in the MASTER table in the IMS DPROP directory.

Table 17 describes the columns in the RUP CBT.

Table 17. Columns of the RUP Control Block Table and Their Data Types

Column	Data type	Description
DBNAME	char(8)	Database name
SEGNAME	char(8)	Segment name
SEQUENCE	smallint	Sequence number
DPRNAME	char(8)	IMS DPROP system name
DPRTOKEN	dec(15,0)	IMS DPROP system token
TYPE	char(1)	Type of the DPRCBT row (0|1|2|3)
TIMESTM	timestamp	PRCB timestamp
VARCB	long varchar	Internal control block

The DPRRCBT columns are:

DBNAME
The name of the IMS database being propagated.

SEGNAME
The name of the segment in the database being propagated.

SEQUENCE
The sequence used for the rows belonging to the same RUP PRCB.

DPRNAME
The name of the IMS DPROP system.

DPRTOKEN
A unique token associated with the DPRNAME.

TYPE
One of the following types of row:

0 Neither the first nor last row of the RUP PRCB

1 The first row (of many) of the RUP PRCB

2 The last row (of many) of the RUP PRCB

3 The first and last (unique) row of the RUP PRCB

TIMESTM
The PRCB creation timestamp.

VARCB
An internal control block that contains information related to a specific database and segment.

The HUP Control Block Table (DPRHCBT)

There is one HUP CBT for each IMS DPROP directory. IMS DPROP generates, deletes or regenerates rows in the RUP CBT when it creates, deletes or recreates propagation requests.

The HUP CBT contains one HUP PRCB for:
- Each TYPE=E propagation request defined in the IMS DPROP directory
- One or more TYPE=U propagation requests having the same table qualifier/table name

During create, delete, or recreate processing for a propagation request, the corresponding rows in the HUP CBT are either generated, deleted, or regenerated.

Table 18 describes the columns in the HUP CBT.

Table 18. Columns of the HUP Control Block Table and Their Data Types

Column	Data type	Description
TABQUAL	char(8)	Table qualifier
TABNAME	varchar(18)	Table name
SEQUENCE	smallint	Sequence number
DPRNAME	char(8)	IMS DPROP system name
DPRTOKEN	dec(15,0)	IMS DPROP system token
TYPE	char(1)	Type of the DPRHCBT row (0|1|2|3)
TIMESTM	timestamp	HUP PRCB timestamp
VARCB	long varchar	Internal control block

TABQUAL
The qualifier of the table from which data is propagated.

TABNAME
The name of the DB2 table from which data is propagated.

SEQUENCE
The sequence used for the rows belonging to the same HUP PRCB.

DPRNAME
The name of the IMS DPROP system.

DPRTOKEN
A unique token associated with the DPRNAME.

TYPE
One of the following types of row:

0 neither the first nor last row of the HUP PRCB

1 the first row (of many) of the HUP PRCB

2 the last row (of many) of the HUP PRCB

3 the first and last (unique) row of the HUP PRCB

TIMESTM
The HUP propagation request timestamp.

VARCB
An internal control block that contains information related to a specific table qualifier and table.

Mapping Tables

The MVG generates the rows of the mapping tables of the IMS DPROP directory. The mapping tables hold the mapping definitions for each propagation request. You can query the tables to obtain information about a propagation request. This topic describes the mapping tables.

The Propagation Request Mapping Table (DPRPR)

The propagation request mapping table is the base table for propagation. It describes the propagation request type, the mapping case, the mapping direction, and the status of the propagation request. There is one row in the mapping table for each propagation request.

The PRID column composes the primary key of the propagation request mapping table. The DPRNAME and DPRTOKEN columns are referred to as a foreign key in the MASTER table in the IMS DPROP directory.

Table 19 describes the columns in the propagation request mapping table.

Table 19. Columns of the Propagation Request Mapping Table and Their Data Types

Column	Data type	Description
PRID	char(8)	Propagation request ID
USERID	char(8)	User ID
NODEID	char(8)	Node ID
PRTYPE	char(1)	Propagation request type (E|L|F|U)
MAPCASE	char(1)	Mapping case (1|2|3)
MAPDIR	char(2)	Mapping direction (HR|TW|RH)
TABQUAL2	char(8)	Table qualifier used for validation
ERROPT	char(1)	Erropt option (B|I)
PRSET	char(8)	Set of propagation requests
PROPSUP	char(1)	Selective suppression of propagation (Y|N)
EXITNAME	char(8)	Propagation exit name (if propagation request type is U)
TIMESTM	timestamp	Propagation request creation timestamp
MAXERROR	integer	Number of error messages
CCUFKEY	char(1)	Primary key mapping indicator used by the CCU
DPRVERS	char(6)	IMS DPROP version
STATUS	char(3)	Propagation request status (ACT|INA|SUS)
XSTATUS	char(1)	Extended status (O|E)
PROPOFF	char(1)	Propagation-off flag (N|Y)
JOBNAME	char(8)	Name of the job authorized to run in PROP-OFF mode
TRACE	smallint	Trace flag / Debugging level (0|1|2|...)
KEYFLAG	char(1)	Parent key flag (internal use)
DPRNAME	char(8)	IMS DPROP system name
DPRTOKEN	dec(15,0)	IMS DPROP system token
WHRFLAG	char(1)	Propagation request has (Y) or does not have (N) a WHERE clause
REVALRES	char(1)	Result of the last revalidation (I|W|E) or blank
REVALTMS	timestamp	Timestamp of the last revalidation, if REVALRES is not blank
BIND	varchar(254)	Bind options for the SQL update module
PATH	char(6)	Kind of path data selected in the propagation request (ID|DENORM|blank)
AVU	char(1)	Avoid unnecessary updates (Y|N|blank)

Table 19. Columns of the Propagation Request Mapping Table and Their Data Types (continued)

Column	Data type	Description
DEFVEXT	char(1)	Default value extension segment (Y|N|blank)
COMMENT	varchar(254)	Comment about the described propagation request
CCD	char(1)	Consistent changed data table
CDNS	char(1)	Condensed CCD table

The DPRPR columns are:

PRID
The propagation request identifier.

USERID
The user identifier.

NODEID
The node identifier.

PRTYPE (E | L | F | U)
A letter that indicates whether the propagation request is extended-function (E), limited function (L), full-function (F), or user mapping (U).

MAPCASE (1 | 2 | 3)
A number that indicates whether the propagation request belongs to generalized mapping case 1, 2, or 3.

Mapping case 1 provides mapping between IMS data of one entity segment and its concatenated key to one relational table.

Mapping case 2 provides mapping between IMS data of one entity segment and its concatenated key and one or more extension segments, to or from one relational table.

Mapping case 3 provides mapping between data of one internal segment (the entity segment) and its concatenated key to one relational table.

For user mapping, the value in this column has no effect on IMS DPROP.

MAPDIR (HR|TW|RH)
The direction in which the data is to be propagated.

HR Hierarchical to relational

TW Hierarchical to relational and relational to hierarchical

RH Relational to hierarchical

TABQUAL2
A qualifier used to access the DB2 catalog for validation if the qualifier of the propagated table is not specified.

ERROPT (B | I)
A letter that describes whether propagation failures result in a backout of the unit of work (B) or whether failures are ignored (I).

PRSET
The name of the propagation set in which this propagation request belongs.

PROPSUP (Y | N)
A letter that indicates whether propagation suppression is allowed when a Segment exit routine returns with a return code of 8.

EXITNAME
The name of the Propagation exit routine if PRTYPE=U.

TIMESTM
The propagation request creation timestamp that helps to control integrity of propagation request information.

MAXERROR
The maximum number of error messages written in a 15-minute interval for propagation errors associated with the propagation request if ERROPT=I.

CCUFKEY (Y | N)
A letter that indicates whether the fully concatenated key of the entity segment is entirely mapped to the DB2 primary key of the propagated table.

DPRVERS
The IMS DPROP version. This column is generated by the MVG.

STATUS (ACT | INA | SUS)
The status of the propagation request. The status can be active, inactive, or suspended. This column is initialized by MVG to inactive (INA). You can change this flag with the SCU.

XSTATUS (O | E)
If STATUS=INA, this column indicates whether the propagation request was inactivated in an orderly (O) or emergency (E) manner. It is initialized by MVG to O. You can change this flag with the SCU.

PROPOFF (N | Y)
A letter that indicates whether a job running with PROP OFF can execute. This column is initialized by MVG to N. You can reset this field with the SCU.

JOBNAME
The name of the job authorized to run with PROPOFF. This column is initialized to blank, and can be set with the SCU.

TRACE (0 | 1 | 2 | ...)
The current debugging level (TRACE=1,2,...), or a zero to indicate that no trace is active (TRACE=0). This column is initialized by MVG to 0. You can set this column with the SCU.

KEYFLAG
A flag used to validate the key mapping when MAPCASE=1 and PRTYPE=F. This column is set by the MVG.

DPRNAME
The name of the IMS DPROP system.

DPRTOKEN
A unique token associated with the DPRNAME.

WHRFLAG (Y | N)
An indication whether (Y) or not (N) this propagation request has a WHERE clause specified.

REVALRES (I | W | E |)
The condition code of the last revalidation of the propagation request:

I Information that the last revalidation ended with neither errors nor warnings

W Warnings were issued during last revalidation

E Errors were issued during last revalidation

blank Indicates that this propagation request has not yet been revalidated

REVALTMS
The timestamp of the last revalidation of this propagation request, which is valid only if REVALRES is not blank.

BIND
The bind options provided on the BIND Propagation parameter and used by MVG to create or recreate the package of the SQL update module (generalized mapping case propagation request only).

This column contains the collection ID of where the package of the SQL update module is bound.

PATH (ID | DENORM |)
A value that, if not blank, indicates the kind of non-key path data selected in the propagation request:

ID Indicates that path data fields selected in the propagation request are only IMS ID fields; that is, fields that cannot change their value.

DENORM
Indicates that path data fields selected in the propagation request can change their value and are not mapped to the DB2 primary key of the propagated table.

AVU (Y | N |)
A value that indicates when to perform SQL updates:

Y Replacing an IMS segment results in an SQL update if at least one propagated field has changed.

N The SQL update is always performed, even if no field has changed.

blank
IMS DPROP determines at propagation time whether to use Y or N.

DEFVEXT (Y | N |)
Identifies default value extension segments, which are used for propagation requests with mapping case 2 that perform DB2-to-IMS propagation.

This column tells IMS DPROP what to do with an extension segment during DB2-to-IMS propagation of an SQL insert and an SQL replace (that change at least one column mapped to the extension segment) when all the target columns of an extension segment are either null or have default values. [6]

Y The propagation of the SQL update or insert results in an occurrence of the extension segment type, even if all of its fields are propagated from columns having a NULL or default value.

N The propagation of the SQL update or insert results in zero occurrences of the extension segment type.

blank
For this column, blank has the same value as 'Y'.

COMMENT
Comments about the described propagation request provided via the COMMENT Propagation parameter.

6. For date, time, and timestamp columns, IMS DPROP does not distinguish between default and nondefault values. Therefore, when processing such columns that are not NULL, IMS DPROP assumes that they have a nondefault value.

CCD
A value that indicates whether the target table can be propagated by IMS DPROP. This column is generated by the MVG.

CDNS
A value that indicates whether the consistent changed data table (CCD) used by IMS DPROP is condensed. A condensed CCD table reflects the current state of the DB2 table.

The MSG Mapping Table (DPRMSG)

The MSG mapping table of the IMS DPROP directory can contain warning and error messages issued by MVG when a propagation request is created or revalidated. You can query this table for information if propagation is not successful for a specific propagation request.

For each propagation request, the table can contain zero, one, or more rows. If a propagation request is revalidated at least once, the MSG mapping table can contains one of two sets of messages. One set of messages is issued when the propagation request is created; one set is issued if it revalidated. If the propagation request is revalidated two or more times, the new messages replace the old ones.

The PRID, ORIGIN, MESSAGE, and SEQUENCE columns compose the primary key of the MSG mapping table. The PRID column is referred to as a foreign key in the propagation request mapping table in the IMS DPROP directory. DPRNAME and DPRTOKEN columns are referred to as a foreign key in the MASTER table in the IMS DPROP directory.

The columns in the MSG mapping table are described in Table 20.

Table 20. Columns of the MSG Mapping Table and Their Data Types

Column	Data Type	Description
PRID	char(8)	PRID
MESSAGE	char(8)	Message number
SEQUENCE	smallint	Sequence number
DBNAME	char(8)	Database name
SEGNAME	char(8)	Segment name
TABQUAL	char(8)	Table qualifier
TABNAME	varchar(18)	Table name
DPRNAME	char(8)	IMS DPROP system name
DPRTOKEN	dec(15,0)	IMS DPROP system token
TEXT	varchar(254)	Message text
ORIGIN	char(1)	Message origin (Rlblank)

The DPRMSG columns are:

PRID
The propagation request identifier.

MESSAGE
The message number issued by MVG.

SEQUENCE
A sequence number used to enforce uniqueness of the DB2 primary key of the MSG mapping table.

DBNAME
The names of the database involved in propagation.

SEGNAME
The names of the segment involved in propagation.

TABQUAL
The qualifier of the table to which data is propagated.

TABNAME
The name of the table to which data is propagated.

DPRNAME
The name of the IMS DPROP system.

DPRTOKEN
A unique token associated with the DPRNAME.

TEXT
The message number and the text of the message itself.

ORIGIN (R |)
An indication that the message was issued during a create (blank) or a revalidate (R) function.

The WHR Mapping Table (DPRWHR)

The WHR mapping table of the IMS DPROP directory contains a WHERE clause, which IMS DPROP uses to propagate data selectively. The WHERE clause is optional; it is stored in the WHR mapping table. You can use it to understand the selection criteria for data propagation.

For each propagation request with a WHERE clause, the table contains one or several rows, depending on the size of the WHERE clause. If the WHERE text is longer than 254 bytes, the text can take several rows.

The PRID and SEQUENCE columns compose the primary key of the WHR mapping table. The PRID column is referred to as a foreign key in the propagation request mapping table in the IMS DPROP directory. The DPRNAME and DPRTOKEN columns are referred to as a foreign key in the MASTER table in the IMS DPROP directory.

The columns in the WHR mapping table are described in Table 21.

Table 21. Columns of the WHR Mapping Table and Their Data Types

Column	Data Type	Description
PRID	char(8)	Propagation request ID
SEQUENCE	smallint	Sequence number
WHRTEXT	varchar(254)	At most 254 bytes of the WHERE clause itself
DPRNAME	char(8)	IMS DPROP system name
DPRTOKEN	dec(15,0)	IMS DPROP system token

The following list describes the DPRWHR column names.

PRID
The propagation request identifier.

SEQUENCE
The unique identifier of each row if the WHERE text is longer than 254 bytes and is longer than one row.

WHRTEXT
The WHERE clause as provided at propagation request coding time.

DPRNAME
The name of the IMS DPROP system.

DPRTOKEN
A unique token associated with the DPRNAME.

The SEG Mapping Table (DPRSEG)

The SEG mapping table contains information about segments and data relating to a specific propagation request. The same segment can be processed by different propagation requests. If a segment is processed by different propagation requests it appears multiple times in the SEG mapping table, but with a different propagation request ID (PRID) each time.

For a propagation request that belongs to mapping case 1, the SEG mapping table contains one row for each of the following segments:
- The entity segment
- Each physical parent/ancestor of the entity segment up to the root

For a propagation request that belongs to mapping case 2, the SEG mapping table contains the same rows as mapping case 1, and in addition, a row for each extension segment.

For a propagation request belonging to mapping case 3, the entity segment is an INTERNAL segment. Therefore the SEG mapping table contains one row for each of the following segments:
- The entity segment, which is an internal segment
- The parent containing segment, which is the IMS segment that contains this internal segment
- Each physical parent or ancestor of the containing parent segment up to the root
- Each STARTSEG segment, which is each internal segment preceding the entity segment and used to locate the entity segment.

For a propagation request that belongs to a user mapping case (PRTYPE=U), the SEG mapping table contains one row for each segment propagated by the propagation request.

The PRID, DBNAME, and SEGNAME columns compose the primary key of the SEG mapping table. The PRID column is referred to as a foreign key in the propagation request table in the IMS DPROP directory. The DPRNAME and DPRTOKEN columns are referred to as a foreign key in the MASTER table in the IMS DPROP directory.

Table 22 describes the columns in the SEG mapping table.

Table 22. Columns of the SEG Mapping Table and Their Data Types

Column	Data Type	Description
PRID	char(8)	Propagation request ID
DBNAME	char(8)	Database name
SEGNAME	char(8)	Segment name
PARENT	char(8)	Parent segment name
BYTES	integer	Segment length in bytes
KEYNAME	char(8)	Name of segment's key field
KEYLENG	integer	Length of segment's key field
SEGOFFS	integer	Offset of segment's key field within the segment
KEYOFFS	integer	Offset of segment's key field within the KFBA
FORMAT	char(2)	Segment format (F|V|FI|VI)
LEVEL	smallint	Segment level
ROLE	char(2)	Segment role (E|X|P|C|S)
SEGEXIT	char(8)	Segment exit name
SEGEXITL	integer	Segment length after exit
SEGEXITF	char(2)	Segment format after exit
DBDVERS	varchar(254)	DBD version
DPRNAME	char(8)	IMS DPROP system name
DPRTOKEN	dec(15,0)	IMS DPROP system token
PATHSEL	char(1)	Type of path data selected in this parent segment (I|D|blank)
OCCURS	char(32)	Number of occurrences of this internal segment
START	char(38)	Start position of this internal segment
STARTFLG	char(1)	Type of information provided in START (blank|F|S)
NEXT	char(38)	Next occurrence of this internal segment
MINBYTES	integer	Minimum variable segment length for IMS DPROP

The following list describes the DPRSEG column names.

PRID
The propagation request identifier.

DBNAME
The name of the database involved in propagation.

SEGNAME
The name of the segment involved in propagation.

The described segment can be an IMS segment or an internal segment.

PARENT
The name of the parent of the described segment.

If the described segment is an internal segment, its parent is the IMS segment that contains this internal segment.

BYTES
The length of the described segment.

For fixed-length IMS or internal segments, BYTES contains the accurate length of the segment. For variable-length IMS segments, BYTES contains the maximum segment length. For variable-length internal segments, BYTES is set to zero.

KEYNAME
The name of the key field of the described segment.

KEYLENG
The length of the key field of the described segment.

SEGOFFS
The offset within the segment of the key field of the described segment.

KEYOFFS
The offset of the key field within the key feedback area (KFBA).

FORMAT (F | V | FI | VI)
The segment format.

If the described segment is an IMS segment, it indicates whether the format of the described segment is fixed (F) or variable (V) in length.

If the described segment is an internal segment, FI means that the described segment is a fixed-length internal segment, and VI means that the described segment is a variable-length internal segment.

LEVEL
The hierarchical level of the described segment.

An internal segment has the level of its containing IMS segment + 1.

ROLE (E | X | P | C | S)
A letter that identifies the segment's role in the propagation request.

The segment can be either:

E The entity segment

X An extension segment of the entity segment (mapping case 2)

P A parent or ancestor of the entity segment

C The IMS segment containing the entity segment (with mapping case 3, the entity segment is an internal segment)

S A STARTSEG segment, that is, an internal segment used to locate the entity segment (mapping case 3)

For user mapping, this column is blank.

SEGEXIT
The names of the Segment exit routine.

SEGEXITL
The length of segment output from the Segment exit routine.

SEGEXITF (F | V)
The format of segment output from the Segment exit routine.

If the segment is processed by a Segment exit routine, this column describes the format of the segment the way it appears after processing by the exit: F for fixed-length segments, and V for variable-length segments.

DBDVERS
The IMS DBD version identifier of the DBD. It is used for mapping integrity.

DPRNAME
The name of the IMS DPROP system.

DPRTOKEN
A unique token associated with the DPRNAME.

PATHSEL (I | D |)
The kind of path data selected from this segment.

I IMS ID fields

D Denormalized path data

blank This segment has no path data selected

MINBYTES
The minimum variable segment length used internally by the HUP during an IMS ISRT or IMS REPL.

This column is relevant only for variable-length IMS segments; for fixed-length IMS segments or internal segments, MINBYTES contains zero.

For variable-length segments, MINBYTES contains the minimum variable segment length that IMS DPROP inserts or replaces. However, this length is not necessarily the same as the minimum segment length provided at DBDGEN. The MINBYTES length:
- Is determined by IMS DPROP
- Depends on the type of IMS database
- Depends on the fields defined for this segment in the propagation request

Column Descriptions for Internal Segments

The following columns are for internal segments. They contain values other than null or default only if the described segment is an internal segment (FORMAT=FI or VI).

Table 23 shows which information is provided, depending on the specifications of the internal segment.

Table 23. Internal Segments Description

Internal Segment Specification	Values of the SEG Table Columns	
Segment start position is	fixed	START=n
	variable	START=fldname+n or START=segname+n
Segment length is	fixed	NEXT=fldname+n or BYTES=n
	variable	NEXT=fldname+n
Number of segment occurrences is	fixed	OCCURS=n
	variable	OCCURS=fldname

OCCURS
The number of occurrences of the internal segment within its containing parent.

This column contains either a numeric literal or a field name. If the number of occurrences is fixed, the numeric literal is the number of occurrences of the internal segment.

If the number of occurrences is variable, field name is the name of a field in the containing IMS parent segment that contains the number of occurrences of this described internal segment.

START
The start position of the first occurrence of the internal segment.

If the segment starts in a fixed position, START contains only a numeric literal identifying the start position of the internal segment within its containing IMS parent segment.

If the segment starts in a variable position, START can be:

fieldname+n
Where *fieldname* is the name of the field located in the containing IMS segment that is used to find this internal segment.

The rightmost byte of this field plus the offset (n) give the start position of the first occurrence of this internal segment.

segname+n
Where *segname* is the name of an internal STARTSEG segment preceding this internal segment in the containing IMS segment that is used to find this internal segment.

The rightmost byte of the STARTSEG segment plus the offset (n) give the start position of the first occurrence of this internal segment.

The referenced STARTSEG segment is also defined in a SEG row, with a ROLE of **S**.

STARTFLG (|F|S)
An indication of what the START column contains:

blank
Indicates that START contains only the numeric literal identifying the start position of the internal segment (the field starts in a fixed position)

F Indicates that START contains *fieldname + literal*

S Indicates that START contains *segname + literal*

NEXT
The next occurrence of the internal segment. If BYTES is not specified, this column contains the following:

fieldname+n
Where *fieldname* is the name of a field located in this internal segment and *n*, an offset value.

The rightmost byte of this field plus the offset value of *n* give the start position of the next segment occurrence.

The TAB Mapping Table (DPRTAB)

The TAB mapping table describes DB2 propagated tables.

If you are using your own mapping cases, you can have one to *n* propagated tables for each propagation request.

If you are using generalized mapping case 1, 2, or 3, you can have more than one propagation request propagate to the same table. However, each propagation request should be bound into a different plan that specifies a unique table identifier.

The PRID, TABQUAL, and TABNAME compose the primary key of the TAB mapping table. The PRID column is referred to as a foreign key in the propagation

request mapping table in the IMS DPROP directory. The DPRNAME and DPRTOKEN columns are referred to as a foreign key in the MASTER table in the IMS DPROP directory.

The columns in the TAB mapping table are described in Table 24.

Table 24. Columns of the TAB Mapping Table and Their Data Types

Column	Data Type	Description
PRID	char(8)	Propagation request ID
TABQUAL	char(8)	Table qualifier
TABNAME	varchar(18)	Table name
DPRNAME	char(8)	IMS DPROP system name
DPRTOKEN	dec(15,0)	IMS DPROP system token

The DPRTAB columns are:

PRID
The propagation request identifier.

TABQUAL
The qualifier of the table to or from which data is propagated.

TABNAME
The name of the table to or from which data is propagated.

DPRNAME
The name of the IMS DPROP system.

DPRTOKEN
A unique token associated with the DPRNAME.

The FLD Mapping Table (DPRFLD)

The FLD mapping table contains information on the fields involved in propagation, and the columns to or from which the fields map.

The FLD mapping table contains one row for each involved field. If a field is involved in multiple propagation requests, it appears multiple times in the FLD mapping table. Each time it appears, it is qualified by its PRID.

The FLD mapping table also contains rows for IMS fields defined in the propagation request but not selected for propagation. Therefore, these fields have no corresponding column in the propagated table. The corresponding information in the FLD mapping table (for example, COLNAME, COLTYPE) is set to blank or zero.

The PRID, DBNAME, SEGNAME, TABQUAL, TABNAME, FLDNAME, and COLNAME columns compose the primary key of the FLD mapping table.

The PRID, DBNAME, and SEGNAME columns are referred to as a foreign key in the SEG mapping table in the IMS DPROP directory. The PRID, TABQUAL, and TABNAME columns are referred to as a foreign key in the TAB mapping table in the IMS DPROP directory. The DPRNAME and DPRTOKEN columns are referred to as a foreign key in the MASTER table in the IMS DPROP directory.

Table 25 describes the columns in the FLD mapping table.

Table 25. Columns of the FLD Mapping Table and Their Data Types

Column	Data Type	Description
PRID	char(8)	Propagation request ID
DBNAME	char(8)	Database name
SEGNAME	char(8)	Segment name
FLDNAME	char(32)	Field name
POSITION	integer	Start position in the segment
FLDTYPE	char(1)	Field type (K|S|D|P)
SSAFLD	char(1)	Field is an SSA field mapped to the DB2 primary key (Y|N)
DATATYPE	char(2)	Field data type in IMS DPROP terminology
DATATYP2	char(8)	Field data type in DB2 terminology
BYTES	integer	Length of the field in bytes
LENFIELD	char(32)	Name of the field containing the length of this described field
SCALE	smallint	Scale factor if decimal field
FLDEXIT	char(8)	Field exit name
FLDETYPE	char(2)	Field data type after exit
FLDEBYTE	integer	Field length after exit
FLDESCAL	smallint	Field scale factor after exit
TABQUAL	char(8)	Table qualifier
TABNAME	varchar(18)	Table name
COLNAME	varchar(18)	Column name
COLTYPE	char(8)	Column data type in DB2 terminology
COLTYPD	char(2)	Column data type in IMS DPROP terminology
COLSCALE	smallint	Column scale factor
COLLENG	integer	Column length
COLSEQ	char(1)	Column is primary key (Y|N)
COLORDER	char(1)	Key ordering sequence (A|D)
COLNULL	char(1)	Column nullability (Y|N|W)
SELECTED	char(1)	Field is selected for propagation (Y|N)
PARMOFFS	integer	Field offset in an internal control block
COLNO	smallint	Column sequence number in the propagated table
DPRNAME	char(8)	IMS DPROP system name
DPRTOKEN	dec(15,0)	IMS DPROP system token
INITFLD	char(1)	Field is to be initialized (Y|N|blank)
PARMOFF2	integer	Field offset of the old value for path data fields

The following list describes the DPRFLD column names.

PRID
The propagation request identifier.

DBNAME
The name of the database participating in propagation.

SEGNAME
The name of the segment participating in propagation.

FLDNAME
The name of the field participating in propagation.

POSITION
The start position of the field in the segment. The first byte within the segment start position is one (not zero).

FLDTYPE (K | S | D | P)
The described field type:

- A key field (K).
- A sub-key field (S).
- A data field (D).
- A field used as path data (P) in one or several dependent segments.

  A field can have a value of P only if its segment is a parent segment (segment ROLE = P or C).

SSAFLD (Y | N)
Indicates whether this field is a non-key segment search argument (SSA) field and if it is mapped to the primary key of the propagated table.

DATATYPE
The field data type in IMS DPROP terminology. See Table 25 on page 158 for more information on data types.

DATATYP2
The field data type in DB2 terminology. See the appropriate Administrators Guide for your propagation mode for more information on data types.

BYTES
The field length in bytes.

LENFIELD
The name of the field that contains the length of the described field when the described field has a variable length.

SCALE
The scale factor of the described field.

FLDEXIT
The name of the Field exit routine processing this field, if DATATYPE or DATATYP2 indicates a user-defined data type.

FLDETYPE
The data type of this field after processing by a field exit; it can contain the same values as DATATYPE, except a user-defined data type.

FLDEBYTE
The length in bytes of the described field after processing by the Field exit routine.

FLDESCAL
The scale factor of the described field after processing by the Field exit routine. FLDESCAL can be specified only for numeric fields in extended format.

TABQUAL
Either blank or the qualifier of the table to or from which data is propagated.

TABNAME
The name of the table to or from which data is propagated.

COLNAME
The name of the column in the propagated table to which the described field is mapped. COLNAME is blank for fields that are not propagated.

COLTYPE
The data type of the target column.

COLTYPD
The data type of the target column in IMS DPROP terminology.

COLSCALE
The scale factor of the target column.

COLLENG
The length of the target column.

COLSEQ (Y | N)
Indicates if the column is (Y) or is not (N) part of the primary key.

COLORDER (A | D)
The key ordering sequences, ascending (A) or descending (D), if the column is part of the primary key.

COLNULL (Y | N | W)
An indication of whether the column is defined as nullable (Y), NOT NULL (N), or NOT NULL WITH DEFAULT (W).

SELECTED (Y | N)
An indication that the field is (Y) or is not (N) selected for propagation.

PARMOFFS
The field offset in the DSECT for the SQL update module. Reserved for IMS DPROP.

COLNO
The numeric place of the target column in the table.

DPRNAME
The name of the IMS DPROP system.

DPRTOKEN
A unique token associated with the DPRNAME.

INITFLD (Y | N |)
This column is relevant only if the SELECTED column contains an N.

Y Indicates that during DB2-to-IMS propagation, the described IMS field is initialized to the value corresponding to its data type (blank for character fields, zero for numeric fields, and so on).

N Indicates that during DB2-to-IMS propagation, this IMS field is not initialized with the value corresponding to its data type. Instead, it is set to binary zero.

blank
This column contains a blank if the propagation request is not PRTYPE=E (additionally, the field is not mapped from DB2 to IMS).

For PRTYPE=E propagation requests coded via the MVG input tables, this column always contains a Y.

For PRTYPE=E propagation requests coded via DXT or DataRefresher, this column contains either Y or N, depending on the DXT or DataRefresher release and depending on whether the field is defined in the DXTVIEW or not.

See “CREATE DXTVIEW ” on page 45 for more information about the CREATE DXTVIEW command.

PARMOFF2
The field offset of the old value of the field in the DSECT passed to the SQL update module.

This column is reserved for IMS DPROP. It is relevant only for non-key path data fields.

Control Tables Used by LOG-ASYNC

Refer to the following topics for more information on the tables used by LOG-ASYNC:

- “The Receiver Control Table (DPRRCT) ”
- “The Propagation Request Control Table (DPRPRCT) ” on page 163
- “The PRDS Register Table (DPRPRDSR) ” on page 163
- “The PRDS Volume Table (DPRPRDSV) ” on page 165

The Receiver Control Table (DPRRCT)

The Receiver control table (RCT) is created when you generate a LOG-ASYNC IMS DPROP system. It exists on the Receiver site, and contains information relating to Receiver executions to:

- Provide information for restarts from normal and abnormal terminations.
- Define which receivers can process which propagation groups. This information, combined with the propagation request assignment to receivers in the PRCT, defines which propagation requests make up a propagation group. See the appropriate Administrators Guide for your propagation mode for a description of propagation group concepts.

When the RCT is created, it is empty. You must create a row in the RCT for each receiver before you assign propagation requests to that Receiver. You use SCU CREATEREC control statements to provide the values for the RECNAME and GROUPID columns. The other columns in the row remain empty until the first execution of the Receiver defined in the row.

The RECNAME column composes the primary key of the RCT.

Table 26 shows the columns in the RCT.

Table 26. Columns of the Receiver Control Table and Their Data Types

Column	Data type	Description
RECNAME	char(8)	Receiver name
RECSTAT	char(9)	Receiver status
GROUPID	char(8)	PRDS group
PRDSSEQ	integer	Sequence number
SRCTYPE	char(8)	Source of PRDS
PRDSSTAT	char(6)	Status of PRDS

Table 26. Columns of the Receiver Control Table and Their Data Types (continued)

Column	Data type	Description
UOWID	char(16)	ID of last committed unit of work

The DPRRCT columns are:

RECNAME
The name of the Receiver. This is the name you supply in the Receiver user input and in the SCU CREATEREC and ASSIGNPR control statements.

RECSTAT
The status of a Receiver. The four possible values for this column are:

COMPLETED
Indicates that the Receiver has completed processing and terminated normally.

EXECUTING
Indicates that the Receiver is actively processing a PRDS. This status can also appear when the Receiver terminates abnormally, but the Receiver could not alter the status before it terminated.

An example of abnormal termination is a system crash.

ABORTED
Indicates that the Receiver has terminated abnormally when the status was altered by the Receiver before termination.

blank
Indicates that you created the Receiver row in the RCT via the SCU, but this Receiver did not yet execute.

GROUPID
The PRDS group to be processed by the Receiver. This is the group identifier you specified in the SCU CREATEREC statement.

PRDSSEQ
The sequence number of the PRDS currently being processed. PRDSs are created in sequential order by the Selector for each group.

The header in the PRDS reflects the sequence of the PRDS, as does the entry in the PRDS register table for the PRDS. The Receiver tracks the sequence numbers of the PRDSs processed and applies the PRDS in the correct sequence based on this number.

SRCTYPE
The source of the PRDS.

PRDSSTAT
The status of the PRDS currently being processed. The three possible values for this column are:

OPEN
Indicates that the Receiver started processing a PRDS, but did not complete processing it.

CLOSED
Indicates that the Receiver processed a PRDS completely.

blank
Indicates the status of the Receiver row in the RCT before the first execution of the Receiver.

UOWID
The recovery token of the last source unit of work (UOW) committed in a Receiver DB2 commit. This column is used to track the commit status of UOWs contained in a PRDS. If a failure occurs during processing and the Receiver stops, restart can proceed correctly from the UOW following the last committed UOW.

The Propagation Request Control Table (DPRPRCT)

The propagation request control table (PRCT) exists at the Receiver site. It is created during installation and generation of a LOG-ASYNC IMS DPROP system.

The PRCT records information about which propagation requests are assigned to which Receivers, and prevents duplication of applied updates. A propagation request can be assigned to only one Receiver at a time.

When the PRCT is created, it is empty. Rows are added to the PRCT when the SCU executes ASSIGNPR request control statements.

The PRID and RECNAME columns compose the primary key of the PRCT. The PRID column is referred to as a foreign key in the propagation request mapping table in the IMS DPROP directory. The RECNAME column is referred to as a foreign key in the RCT in the IMS DPROP directory.

You must create a row for a Receiver in the RCT before assigning propagation requests to that Receiver.

Table 27 describes the columns in the PRCT.

Table 27. Columns of the Propagation Request Control Table and Their Data Types

Column	Data type	Description
PRID	char(8)	PRID
RECNAME	char(8)	Receiver name

The DPRPRCT columns are:

PRID
The propagation request identifier.

RECNAME
The names of the Receiver to which the propagation request is assigned.

The PRDS Register Table (DPRPRDSR)

The PRDS register table exists only at the Receiver site. It:
- Catalogs PRDS entries for a Receiver site.
- Informs the Receiver that a PRDS is available to be received.
- Allows the Receiver to dynamically allocate a PRDS data set.

You use the PRDS Registration utility (PRU) to register PRDS entries in the PRDS register table.

For each uncataloged PRDS, the volume table (DPRPRDSV) contains one or more rows.

The GROUPID and PRSSEQ columns compose the primary key of the PRDS register table.

Table 28 describes the columns in the PRDS register table.

Table 28. Columns of the PRDS Register Table and Their Data Types

Column	Data Type	Description
GROUPID	char(8)	PRDS group
PRDSSEQ	integer	Sequence number
SRCTYPE	char(8)	Source of PRDS
PRDSDSN	char(44)	Data set name (DSN) of PRDS
PRDSUNIT	char(8)	Unit type of PRDS
PRDSLAB	char(3)	Label where tape PRDS resides
TAPESEQ	smallint	Positional sequence on tape
REGTSM	timestamp char(26)	Date PRDS was registered by the PRU
DSNQUAL1 to DSNQUAL22	char(8)	Qualifiers for DSN

The DPRPRDSR columns are:

GROUPID
The name of the group to which the PRDS belongs. This name is the group identifier that you specify in the Receiver control statements; it links the Receiver to the PRDS.

PRDSSEQ
The sequence number of the PRDS currently being processed. The Selector creates PRDSs in sequential order for each group. The header in the PRDS reflects the sequence of the PRDS, as does the entry in the PRDS register table. The Receiver tracks the sequence numbers of the PRDSs processed and applies the PRDS in the correct sequence based on this number.

SRCTYPE
The source of the PRDS. The only valid PRDS source type for this release is IMS310.

PRDSDSN
Contains the fully qualified data set name of the PRDS.

PRDSUNIT
The unit type where the PRDS resides.

PRDSLAB
The tape label where the tape PRDS resides.

TAPESEQ
The positional sequence of the PRDS on the tape.

REGTSM
The date when the PRDS is registered by the PRU. This value is used to determine the age of a PRDS when it is being unregistered or listed.

DSNQUAL1 to DSNQUAL22
The qualifiers for the data set name of the PRDS.

The PRDS Volume Table (DPRPRDSV)

The PRDS volume table exists only on the Receiver site. It is an extension of the PRDS register table DPRPRDSR. If the DSN is uncataloged, it contains a row for each volume specified on the REGISTER control statement.

The PRDSDSN and PRDSVOL columns comprise the primary key of the PRDS volume table.

Table 29 describes the columns in the PRDS volume table.

Table 29. Columns of the Volume Table and Their Data Types

Column	Data type	Description
PRDSDSN	char(44)	Data set Name (DSN) of PRDS
PRDSVOL	char(6)	Serial number of the volume where PRDS resides

The DPRPRDSV columns are:

PRDSDSN
The fully qualified data set name of the PRDS.

PRDSVOL
The serial number of the volume where the PRDS resides.

Examples of How to Query IMS DPROP Tables

This topic shows examples of queries you can make on IMS DPROP tables.

Example 1 - Status

The example query in Figure 54 produces an overview of the status of all propagation requests, with information on other selected options, sorted by status.

```
SELECT STATUS, XSTATUS, PRID, PRSET,
       ERROPT, PROPOFF, TRACE
  FROM prefix.DPRPR
 ORDER BY
       STATUS, XSTATUS, PRID
```

Figure 54. Example 1: IMS DPROP Table Query

Figure 55 on page 166 shows the report that is produced by the query in Figure 54.

STATUS	XSTATUS	PRID	PRSET	ERROPT	PROPOFF		TRACE
ACT	0	ATD0000	PRSET1	I	Y		0
	0	ATD1000	PRSET1	I	Y		0
	0	ATD1100	PRSET1	I	Y		0
	0	ATD1200	PRSET1	I	Y		0
	0	ATD1210	PRSET1	I	Y		0
	0	ATD2000	PRSET1	I	Y		0
	0	ATD2100	PRSET1	I	Y		0
	0	ATD2110	PRSET1	I	Y		0
	0	ATD3000	PRSET1	I	Y		0
	0	ATD4000	PRSET1	I	Y		0
	0	ATD5000	PRSET1	I	Y		0
	0	ATE0000	PRSET3	I	N		0
	0	ATE1000	PRSET3	I	N		0
	0	ATE1100	PRSET3	I	N		0
	0	ATE1200	PRSET3	I	N		0
	0	ATE1210	PRSET3	I	N		0
	0	ATE2000	PRSET3	I	N		0
	0	ATE3000	PRSET3	I	N		0
	0	ATE4000	PRSET3	I	N		0
	0	ATE5000	PRSET3	I	N		0
	0	ATH0000	PRSET3	I	Y		0
	0	ATH1000	PRSET3	I	Y		0
	0	ATH1100	PRSET3	I	N		0
	0	ATH1200	PRSET3	I	N		0
	0	ATH1210	PRSET3	I	N		0
	0	ATH2000	PRSET3	I	N		0
	0	ATH2100	PRSET3	I	N		0
	0	ATH2110	PRSET3	I	N		0
	0	ATH3000	PRSET3	I	N		0
	0	ATH4000	PRSET3	I	N		0
	0	ATH5000	PRSET3	I	N		0
	0	ATI0000	PRSET2	I	Y		0
	0	ATI1000	PRSET2	I	Y		0
	0	ATI1100	PRSET2	I	N		0
	0	ATI1200	PRSET2	I	N		0
	0	ATI1210	PRSET2	I	N		0
	0	ATI2000	PRSET2	I	N		0
	0	ATI2100	PRSET2	I	N		0
	0	ATI2110	PRSET2	I	N		0
	0	ATI3000	PRSET2	I	N		0
	0	ATI4000	PRSET2	I	N		0
	0	ATI5000	PRSET2	I	N		0
	0	ATS0000	PRSET2	I	N		0
	0	PRO01	PRSET1	I	N		0
SUS	0	ATD1000F	PRSET1	B	N		0

Figure 55. Report for Example 1: IMS DPROP Table Query

Example 2–Cross Reference Between IMS Data and DB2 Tables

The query in Figure 56 on page 167 produces a cross reference between propagated IMS data (DBNAME and SEGNAME) and DB2 tables. The PRID with STATUS and other selected options are also included in the report, which is sorted by DBNAME and SEGNAME.

The WHERE clause excludes rows with ROLE=P because these rows do not describe propagated segments. They describe the parents or ancestors of the propagated segments.

```
SELECT DBNAME, SEGNAME, SEG.PRID, ROLE
       TABQUAL, TABNAME,
       STATUS, XSTATUS, ERROPT, PROPOFF, TRACE
  FROM prefix.DPRSEG SEG,
       prefix.DPRTAB TAB,
       prefix.DPRPR PR
 WHERE SEG.PRID = TAB.PRID
   AND SEG.PRID = PR.PRID
   AND ROLE <> 'P'
 ORDER BY
       DBNAME, SEGNAME
```

Figure 56. Example 2: IMS DPROP Table Query

Figure 57 on page 168 shows the report that is produced by the query in Figure 56.

DBNAME	SEGNAME	PRID	ROLE	TABQUAL	TABNAME	STATUS	XSTATUS	ERROPT	PROPOFF	TRACE
DB1	SEG1	PR001	E		TABLE01	ACT	0	I	N	0
										*
TDBDPD	SD0000	ATD0000	E	DPRPREF	TD0000	ACT	0	I	Y	0
	SD1000	ATD1000F	E	DPRPREF	TD1000F	SUS	0	B	N	0
	SD1000	ATD1000	E	DPRPREF	TD1000	ACT	0	I	Y	0
	SD1100	ATD1100	E	DPRPREF	TD1100	ACT	0	I	Y	0
	SD1200	ATD1200	E	DPRPREF	TD1200	ACT	0	I	Y	0
	SD1210	ATD1210	E	DPRPREF	TD1210	ACT	0	I	Y	0
	SD1300	ATD1000	X	DPRPREF	TD1000	ACT	0	I	Y	0
	SD1400	ATD1000	X	DPRPREF	TD1000	ACT	0	I	Y	0
	SD2000	ATD2000	E	DPRPREF	TD2000	ACT	0	I	Y	0
	SD2100	ATD2100	E	DPRPREF	TD2100	ACT	0	I	Y	0
	SD2110	ATD2110	E	DPRPREF	TD2110	ACT	0	I	Y	0
	SD3000	ATD3000	E	DPRPREF	TD3000	ACT	0	I	Y	0
	SD4000	ATD4000	E	DPRPREF	TD4000	ACT	0	I	Y	0
	SD5000	ATD5000	E	DPRPREF	TD5000	ACT	0	I	Y	0
										0
TDBDPE	SE0000	ATE0000	E	DPRPREF	TE0000	ACT	0	I	N	0
	SE1000	ATE1000	E	DPRPREF	TE1000	ACT	0	I	N	0
	SE1100	ATE1100	E	DPRPREF	TE1100	ACT	0	I	N	0
	SE1200	ATE1200	E	DPRPREF	TE1200	ACT	0	I	N	0
	SE1210	ATE1210	E	DPRPREF	TE1210	ACT	0	I	N	0
	SE1300	ATE1000	X	DPRPREF	TE1000	ACT	0	I	N	0
	SE1400	ATE1000	X	DPRPREF	TE1000	ACT	0	I	N	0
	SE2000	ATE2000	E	DPRPREF	TE2000	ACT	0	I	N	0
	SE3000	ATE3000	E	DPRPREF	TE3000	ACT	0	I	N	0
	SE4000	ATE4000	E	DPRPREF	TE4000	ACT	0	I	N	0
	SE5000	ATE5000	E	DPRPREF	TE5000	ACT	0	I	N	0
TDBDPH	SH0000	ATH0000	E	DPRPREF	TH0000	ACT	0	I	N	0
	SH1000	ATH1000	E	DPRPREF	TH1000	ACT	0	I	N	0
	SH1100	ATH1100	E	DPRPREF	TH1100	ACT	0	I	N	0
	SH1200	ATH1200	E	DPRPREF	TH1200	ACT	0	I	N	0
	SH1210	ATH1210	E	DPRPREF	TH1210	ACT	0	I	N	0
	SH1300	ATH1000	X	DPRPREF	TH1000	ACT	0	I	N	0
	SH1400	ATH1000	X	DPRPREF	TH1000	ACT	0	I	N	0
	SH2000	ATH2000	E	DPRPREF	TH2000	ACT	0	I	N	0
	SH2100	ATH2100	E	DPRPREF	TH2100	ACT	0	I	N	0
	SH2110	ATH2110	E	DPRPREF	TH2110	ACT	0	I	N	0
	SH3000	ATH3000	E	DPRPREF	TH3000	ACT	0	I	N	0
	SH4000	ATH4000	E	DPRPREF	TH4000	ACT	0	I	N	0
	SH5000	ATH5000	E	DPRPREF	TH5000	ACT	0	I	N	0

Figure 57. Report for Example 2: IMS DPROP Table Query

Example 3–Cross Reference Between IMS Data and DB2 Tables

The query in Figure 58 on page 169 produces a cross-reference between propagated IMS data (DBNAME and SEGNAME) and DB2 tables. It also includes the PRID with STATUS and other selected options. The report is sorted by TABQUAL and TABNAME.

The WHERE clause excludes rows with ROLE=P because these rows do not describe propagated segments. They describe the parents or ancestors of the propagated segments.

```
SELECT TABQUAL, TABNAME, TAB.PRID,
       DBNAME,  SEGNAME, ROLE
       STATUS, XSTATUS, ERROPT, PROPOFF, TRACE
  FROM prefix.DPRTAB TAB,
       prefix.DPRSEG SEG,
       prefix.DPRPR PR
 WHERE TAB.PRID = SEG.PRID
   AND TAB.PRID = PR.PRID
   AND ROLE <> 'P'
 ORDER BY
       TABQUAL, TABNAME
```

Figure 58. Example 3: IMS DPROP Table Query

Figure 59 on page 170 shows the report that is produced by the query in Figure 58.

TABQUAL	TABNAME	PRID	DBNAME	SEGNAME	ROLE	STATUS	XSTATUS	ERROPT	PROPOFF	TRACE
	TABLE01	PR001	DB1	SEG1	E	ACT	O	I	N	0
										*
DPRPREF	TD0000	ATD0000	TDBDPD	SD0000	E	ACT	O	I	Y	0
	TD1000	ATD1000	TDBDPD	SD1000	E	ACT	O	I	Y	0
	TD1000	ATD1000	TDBDPD	SD1300	X	ACT	O	I	Y	0
	TD1000	ATD1000	TDBDPD	SD1400	X	ACT	O	I	Y	0
	TD1000F	ATD1000F	TDBDPD	SD1000	E	SUS	O	B	N	0
	TD1100	ATD1100	TDBDPD	SD1100	E	ACT	O	I	Y	0
	TD1200	ATD1200	TDBDPD	SD1200	E	ACT	O	I	Y	0
	TD1210	ATD1210	TDBDPD	SD1210	E	ACT	O	I	Y	0
	TD2000	ATD2000	TDBDPD	SD2000	E	ACT	O	I	Y	0
	TD2100	ATD2100	TDBDPD	SD2100	E	ACT	O	I	Y	0
	TD2110	ATD2110	TDBDPD	SD2110	E	ACT	O	I	Y	0
	TD3000	ATD3000	TDBDPD	SD3000	E	ACT	O	I	Y	0
	TD4000	ATD4000	TDBDPD	SD4000	E	ACT	O	I	Y	0
	TD5000	ATD5000	TDBDPD	SD5000	E	ACT	O	I	Y	0
	TE0000	ATE0000	TDBDPE	SE0000	E	ACT	O	I	N	0
	TE1000	ATE1000	TDBDPE	SE1000	E	ACT	O	I	N	0
	TE1000	ATE1000	TDBDPE	SE1300	X	ACT	O	I	N	0
	TE1000	ATE1000	TDBDPE	SE1400	X	ACT	O	I	N	0
	TE1100	ATE1100	TDBDPE	SE1100	E	ACT	O	I	N	0
	TE1200	ATE1200	TDBDPE	SE1200	E	ACT	O	I	N	0
	TE1210	ATE1210	TDBDPE	SE1210	E	ACT	O	I	N	0
	TE2000	ATE2000	TDBDPE	SE2000	E	ACT	O	I	N	0
	TE3000	ATE3000	TDBDPE	SE3000	E	ACT	O	I	N	0
	TE4000	ATE4000	TDBDPE	SE4000	E	ACT	O	I	N	0
	TE5000	ATE5000	TDBDPE	SE5000	E	ACT	O	I	N	0
	TH0000	ATH0000	TDBDPH	SH0000	E	ACT	O	I	N	0
	TH1000	ATH1000	TDBDPH	SH1000	E	ACT	O	I	N	0
	TH1000	ATH1000	TDBDPH	SH1300	X	ACT	O	I	N	0
	TH1000	ATH1000	TDBDPH	SH1400	X	ACT	O	I	N	0
	TH1100	ATH1100	TDBDPH	SH1100	E	ACT	O	I	N	0
	TH1200	ATH1200	TDBDPH	SH1200	E	ACT	O	I	N	0
	TH1210	ATH1210	TDBDPH	SH1210	E	ACT	O	I	N	0
	TH2000	ATH2000	TDBDPH	SH2000	E	ACT	O	I	N	0
	TH2100	ATH2100	TDBDPH	SH2100	E	ACT	O	I	N	0
	TH2110	ATH2110	TDBDPH	SH2110	E	ACT	O	I	N	0
	TH3000	ATH3000	TDBDPH	SH3000	E	ACT	O	I	N	0
	TH4000	ATH4000	TDBDPH	SH4000	E	ACT	O	I	N	0
	TH5000	ATH5000	TDBDPH	SH5000	E	ACT	O	I	N	0
	TI0000	ATI0000	TDBDPI	SI0000	E	ACT	O	I	N	0
	TI1000	ATI1000	TDBDPI	SI1000	E	ACT	O	I	N	0
	TI1000	ATI1000	TDBDPI	SI1300	X	ACT	O	I	N	0
	TI1000	ATI1000	TDBDPI	SI1400	X	ACT	O	I	N	0
	TI1100	ATI1100	TDBDPI	SI1100	E	ACT	O	I	N	0
	TI1200	ATI1200	TDBDPI	SI1200	E	ACT	O	I	N	0
	TI1210	ATI1210	TDBDPI	SI1210	E	ACT	O	I	N	0
	TI2000	ATI2000	TDBDPI	SI2000	E	ACT	O	I	N	0
	TI2100	ATI2100	TDBDPI	SI2100	E	ACT	O	I	N	0
	TI2110	ATI2110	TDBDPI	SI2110	E	ACT	O	I	N	0
	TI3000	ATI3000	TDBDPI	SI3000	E	ACT	O	I	N	0
	TI4000	ATI4000	TDBDPI	SI4000	E	ACT	O	I	N	0
	TI5000	ATI5000	TDBDPI	SI5000	E	ACT	O	I	N	0
	TS0000	ATS0000	TDBDPS	SS0000	E	ACT	O	I	N	0

Figure 59. Report for Example 3: IMS DPROP Table Query

Example 4–Data Propagated by One Request

The query in Figure 60 shows all fields of IMS data and columns of DB2 data propagated by one propagation request, PRID=ATD3000.

```
SELECT DBNAME,   SEGNAME, FLDNAME,
       POSITION, FLDTYPE, SSAFLD,
       TABQUAL,  TABNAME, COLNAME,
       COLTYPE,  COLSEQ,  COLNO
  FROM prefix.DPRFLD
 WHERE PRID = 'ATD3000'
 ORDER BY
       COLNO
```

Figure 60. Example 4: IMS DPROP Table Query

Figure 61 on page 172 shows the report that is produced by the query in Figure 60.

DBNAME	SEGNAME	FLDNAME	POSITION	FLDTYPE	SSAFLD	TABQUAL	TABNAME	COLNAME	COLTYPE	COLSEQ	COLNO
TDBDPD	SD0000	FTCNBR	1	K	N	DPRPREF	TD3000	CTCNBR	CHAR	Y	1
											*
	SD3000	FSEQNBR1	7	K	N	DPRPREF	TD3000	CSEQNBR1	CHAR	Y	2
			0	D		DPRPREF	TD3000	CSEQNBR2	CHAR	N	3
			0	D		DPRPREF	TD3000	CSEQNBR3	CHAR	N	4
			0	D		DPRPREF	TD3000	CSEQNBR4	CHAR	N	5
		FGENERAL	15	D	N	DPRPREF	TD3000	CGENERAL	CHAR	N	6
		FSBYT1	35	D	N	DPRPREF	TD3000	CSINT1	SMALINT	N	7
		FSBYT1	35	D	N	DPRPREF	TD3000	CINTE1	INTEGER	N	8
		FSBYT1	35	D	N	DPRPREF	TD3000	CDECI1	DECIMAL	N	9
		FSBYT2	36	D	N	DPRPREF	TD3000	CDECI2	DECIMAL	N	10
		FSINT1	37	D	N	DPRPREF	TD3000	CSINT2	SMALINT	N	11
		FSINT1	37	D	N	DPRPREF	TD3000	CINTE2	INTEGER	N	12
		FSINT1	37	D	N	DPRPREF	TD3000	CDECI3	DECIMAL	N	13
		FSINT2	39	D	N	DPRPREF	TD3000	CDECI4	DECIMAL	N	14
		FINTE1	41	D	N	DPRPREF	TD3000	CINTE3	INTEGER	N	15
		FINTE1	41	D	N	DPRPREF	TD3000	CDECI5	DECIMAL	N	16
		FINTE2	45	D	N	DPRPREF	TD3000	CSINT3	SMALINT	N	17
		FINTE2	45	D	N	DPRPREF	TD3000	CDECI6	DECIMAL	N	18
		FPACK1	49	D	N	DPRPREF	TD3000	CSINT4	SMALINT	N	19
		FPACK1	49	D	N	DPRPREF	TD3000	CINTE4	INTEGER	N	20
		FPACK1	49	D	N	DPRPREF	TD3000	CDECI7	DECIMAL	N	21
		FPACK1	49	D	N	DPRPREF	TD3000	CDECI8	DECIMAL	N	22
		FPACK2	50	D	N	DPRPREF	TD3000	CSINT5	SMALINT	N	23
		FPACK2	50	D	N	DPRPREF	TD3000	CINTE5	INTEGER	N	24
		FPACK2	50	D	N	DPRPREF	TD3000	CDECI9	DECIMAL	N	25
		FPACK3	52	D	N	DPRPREF	TD3000	CDECI10	DECIMAL	N	26
		FPACK4	68	D	N	DPRPREF	TD3000	CSINT6	SMALINT	N	27
		FPACK4	68	D	N	DPRPREF	TD3000	CINTE6	INTEGER	N	28
		FPACK4	68	D	N	DPRPREF	TD3000	CDECI11	DECIMAL	N	29
		FZONE1	74	D	N	DPRPREF	TD3000	CSINT7	SMALINT	N	30
		FZONE1	74	D	N	DPRPREF	TD3000	CINTE7	INTEGER	N	31
		FZONE1	74	D	N	DPRPREF	TD3000	CDECI12	DECIMAL	N	32
		FZONE1	74	D	N	DPRPREF	TD3000	CDECI13	DECIMAL	N	33
		FZONE2	75	D	N	DPRPREF	TD3000	CINTE8	INTEGER	N	34
		FZONE2	75	D	N	DPRPREF	TD3000	CDECI14	DECIMAL	N	35
		FZONE2	75	D	N	DPRPREF	TD3000	CDECI15	DECIMAL	N	36
		FZONE3	84	D	N	DPRPREF	TD3000	CDECI16	DECIMAL	N	37
		FZONE3	84	D	N	DPRPREF	TD3000	CDECI17	DECIMAL	N	38
		FZONE4	100	D	N	DPRPREF	TD3000	CSINT8	SMALINT	N	39
		FZONE4	100	D	N	DPRPREF	TD3000	CINTE9	INTEGER	N	40
		FZONE4	100	D	N	DPRPREF	TD3000	CDECI18	DECIMAL	N	41
			0	D		DPRPREF	TD3000	CSFLO1	FLOAT	N	42
			0	D		DPRPREF	TD3000	CDFLO1	FLOAT	N	43
			0	D		DPRPREF	TD3000	CSFLO2	FLOAT	N	44
			0	D		DPRPREF	TD3000	CDFLO2	FLOAT	N	45
			0	D		DPRPREF	TD3000	CSFLO3	FLOAT	N	46

Figure 61. Report for Example 4: IMS DPROP Table Query

Part 4. Synchronous Utilities

Chapter 7. Status Change Utility (SCU) for Synchronous Propagation

This topic describes the use of the Status Change Utility (SCU) for Synchronous propagation. The SCU can also be used for asynchronous MQ-based propagation (MQ-ASYNC); see Chapter 17, “Status Change Utility (SCU) for MQ-ASYNC ,” on page 401. It can also be used for asynchronous log-based (LOG-ASYNC) propagation; see Chapter 8, “Status Change Utility (SCU) for LOG-ASYNC ,” on page 213.

For synchronous propagation, use the Status Change utility (SCU) to control individual propagation requests and to give you limited control over IMS and DB2 databases. The SCU is your primary tool for controlling and operating your IMS DPROP system.

Recommendation: It is strongly recommended that you implement DBRC share control and register the propagated IMS databases used for production. Such full-function databases can be set to read-only mode by using SCU control statements, thus enabling SCU support for orderly status changes.

The following topics provide additional information:
- “Input and Output”
- “JCL Requirements ” on page 176
- “Use of the SCU for Synchronous Propagation ” on page 180
- “States of IMS DPROP Systems and Propagation Requests ” on page 181
- “Control statements for synchronous propagation” on page 188
- “Output Messages and Return Codes of the SCU” on page 206

Input and Output

Figure 62 on page 176 illustrates the input to, and the output created by the SCU.

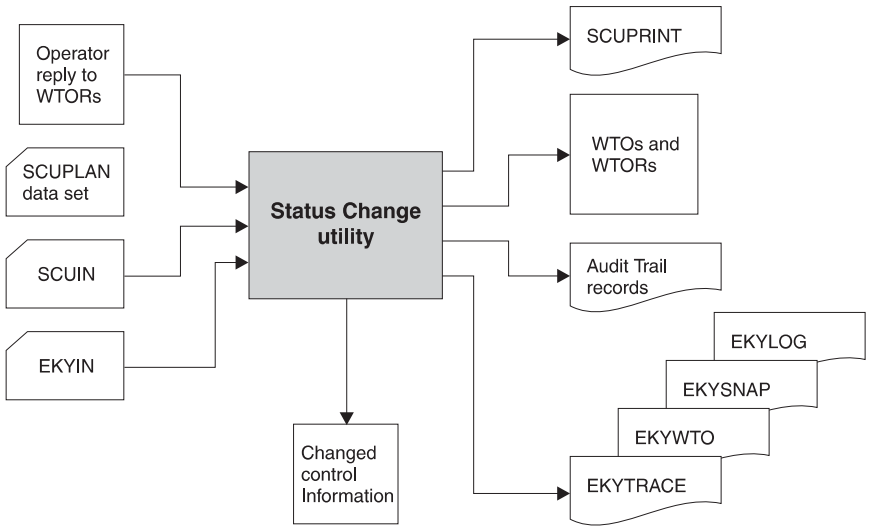

Figure 62. Status Change Utility: Input and Output

The input to the SCU is:

- The //SCUIN data set, which contains the control statements
- The //SCUPLAN data set
- Operator reply to WTORs

  If the SCU is in a wait state because of conflicting database authorizations, the operator can use WTOR to:
  - Enter LIST to list the conflicting authorizations.
  - Enter TERM to terminate execution of the SCU.
- The //EKYIN data set

The output from the SCU is:

- Changed control information:
  - in the DPROP Directory Tables
  - in the DPROP Status file
  - The virtual lookaside facility (VLF) copy of IMS DPROP information
  - The access mode of IMS databases in RECON data sets
- The //EKYTRACE, //EKYWTO, //EKYSNAP and //EKYLOG files
- Audit trail records, see *IMS DPROP Messages and Codes* for listings

JCL Requirements

This topic contains samples of the JCL that executes the SCU.

EKYUSCUP—JCL to Execute the SCU

Figure 63 on page 177 is a sample JCL procedure EKYUSCUP used to execute the SCU.

```
//*-------------------------------------------------------------------*
//*    PROCEDURE EKYUSCUP TO EXECUTE THE SCU                          *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************

//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S NEED:          *
//*  -----------------------------------------------------            *
//*                                                                   *
//*  1111111   - DPROP RESLIB HLQ(S)                                  *
//*                                                                   *
//*-------------------------------------------------------------------*
//EKYUSCU      PROC EKYPREF=1111111
//*-------------------------------------------------------------------*
//SCU          EXEC     PGM=EKYSCU00
//*******************************************************************
//* NOTE: IN ASYNCH MODE, A STATUS FILE IS NOT REQUIRED AND THE       *
//* FOLLOWING DD STATEMENT SHOULD BE REMOVED.                         *
//*                                                                   *
//EKYSTATF  DD   DSN=&EKYPREF..EKYSTATF,DISP=SHR
//*******************************************************************
//*
//EKYRESLB     DD       DISP=SHR,
//             DSN=&EKYPREF..EKYRESLB
//SCUPRINT     DD       SYSOUT=*
//SYSPRINT     DD       DSN=&&SYSPRINT,UNIT=SYSDA,SPACE=(CYL,1)
//SYSIN        DD       DSN=&&SYSIN,UNIT=SYSDA,SPACE=(CYL,1)
//*-------------------------------------------------------------------*
//*******************************************************************
//*-----------------------------------------------------------------
//*       ADD THE RECON DD STATEMENTS HERE IF YOU ARE NOT USING
//*       DYNAMIC ALLOCATION
//*----------------------------------------------------------------
//*       &RECON1   = --+
//*       &RECON2   =   '  -  DBRC RECON DATASETS
//*       &RECON3   = --+
//*
//*******************************************************************
//*-----------------------------------------------------------------
```

Figure 63. Sample JCL Procedure EKYUSCUP

Before you submit the JCL, replace 1111111 with IMS DPROP RESLIB high-level qualifiers.

The DD statement //EKYRESLB is common to many IMS DPROP components and is described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The remaining DD statements used in the EKYUSCUP JCL are:

//EKYSTATF DD Statement
Describes the status of the IMS DPROP system.

If you call the SCU with an INIT DPROP or INIT STATF control statement, the //EKYSTATF DD statement should be either preallocated in a previous job step (for example, using IEFBR14) or allocated in the SCU job step. Figure 64 shows a sample DD statement to allocate the status file on DASD.

```
//EKYSTATF  DD  DSN=DPROP.STATF,DISP=(,CATLG),UNIT=SYSDA,SPACE=(TRK,1)
```

Figure 64. Sample JCL for //EKYSTATF

If the SCU is called for a function other than INIT DPROP and INIT STATF, use the //EKYSTATF statement.

//SCUPRINT
Contains the print output from the SCU. The print output includes a list of all Status Change Utility control statements, confirmation that they have been executed, and a list of SCU error messages.

//SYSPRINT
Contains the print output of the IMS DBRC utility called by the SCU.

This DD statement must be allocated to a temporary data set and *not* allocated as a SYSOUT file.

//SYSIN
Contains input commands generated by the SCU to the IMS DBRC utility. The SCU calls the IMS DBRC utility.

This DD statement must be allocated to a temporary data set and *not* allocated as a SYSIN file.

The SCU uses DBRC and therefore requires access to the DBRC RECON data sets. If you use DBRC without dynamic allocation, you must add the following DD statements to the JCL.

```
//RECON1 DD DSN=recon1,DISP=SHR
//RECON2 DD DSN=recon2,DISP=SHR
//RECON3 DD DSN=recon3,DISP=SHR
```

EKYUSCUJ—JCL to Execute EKYUSCUP

Figure 65 on page 179 is an example of EKYUSCUP, the JCL used to call the EKYUSCUP JCL procedure.

```
//JJJJJJJ  JOB  *** VALID JOB CARD DETAILS ***
//*-------------------------------------------------------------------*
//*    JOB TO EXECUTE THE SCU PROCEDURE (EKYUSCUP)                    *
//*                                                                   *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S NEED:          *
//*  -----------------------------------------------------            *
//*                                                                   *
//*  JJJJJJJ   - JOB NAME                                             *
//*  1111111   - SCU CONTROL STATEMENT(S) AS REQUIRED                 *
//*  2222222   - DB2 SUBSYSTEM NAME                                   *
//*  3333333   - NAME OF THE SCU PLAN                                 *
//*                                                                   *
//*-------------------------------------------------------------------*
//EKYUSCU      EXEC EKYUSCUP
//SCUIN     DD   *
1111111
/*
//SCUPLAN   DD   *
SYSTEM=2222222,PLAN=3333333;
//*-------------------------------------------------------------------*
```

Figure 65. EKYUSCUJ: Sample JCL to Run the Status Change Utility

//SCUPLAN DD Statement

Identifies the file that contains a control statement that identifies the name of the DB2 plan that executes the Status Change Utility. The //SCUPLAN file is input to the Status Change Utility.

Figure 66 shows the syntax of the //SCUPLAN file.

```
►►──PLAN=plan_name──┬──────────────────────┬──;──────────────────────────►◄
                    └─,SYSTEM=system_name──┘
```

Figure 66. The //SCUPLAN file

PLAN= *plan_name*
Names the DB2 plan that executes the SCU.

SYSTEM= *system_name*
Identifies the DB2 subsystem. If you do not use this keyword, the SCU uses the default DB2 subsystem identifier in module DSNHDECP.

Figure 67 on page 180 shows two examples of the //SCUPLAN control statement. The first example shows that DB2 plan TEST01 executes the SCU

on the subsystem identified in DSNHDECP. The second example shows that DB2 plan TEST01 executes the SCU on system TEST02.

```
PLAN=TEST01
PLAN=TEST02,SYSTEM=TEST02;
```

Figure 67. //SCUPLAN control statement Examples

//SCUIN
Contains the stream of Status Change Utility control statements.

Use of the SCU for Synchronous Propagation

In synchronous propagation the SCU is used to change the status of a IMS DPROP system, of individual propagation requests and groups of propagation requests, and of IMS and DB2 databases.

For a IMS DPROP system, the SCU can:
- Set an emergency stop of all propagation activities in a IMS DPROP system.
- Set a non-emergency stop of propagation activities in a IMS DPROP system.
- Display whether the IMS DPROP system is emergency stopped.

For propagation requests, the SCU can:
- Activate, deactivate and suspend propagation requests.
- Emergency deactivate propagation requests.
- Change the error option of a propagation request.

For IMS and DB2 databases, the SCU can:
- Set and reset the read-only status of DB2 databases.
- Allow and deny execution of job steps that include PROP OFF control statements in the //EKYIN data set.
- Set tracing on and off for selected IMS databases or segment types.
- Change control information used by the error-handling logic of RUP and HUP.
- Initialize the master table of the IMS DPROP directory and the status file during customization.
- Build or initialize the VLF objects containing IMS DPROP control information.
- Rebuild the status file is emergency stopped.

Table 30 lists the SCU control statements for synchronous propagation and the areas affected by the control statements.

Table 30. List of Status Changes

Control Statement	Area affected by the Control Statement	Where the Status of the Area is Recorded	Status After the Control Statement is Processed
ACTIVATE	PR	IMS DPROP directory	Active
DEACTIVATE	PR	IMS DPROP directory	Inactive
EDEACTIVATE	PR	IMS DPROP directory	Inactive

Table 30. List of Status Changes (continued)

Control Statement	Area affected by the Control Statement	Where the Status of the Area is Recorded	Status After the Control Statement is Processed
SUSPEND	PR	IMS DPROP directory	Suspended
RESET	PR	IMS DPROP directory	Inactive
RESET	IMS DPROP system	Status file	Not emergency stopped
ESTOP	IMS DPROP system	Status file	Emergency stopped
READON	IMS database DB2 table space/database	RECON data sets DB2	Read-only Read-only
READOFF	IMS database DB2 table space/database	RECON data sets DB2	Read-only off Read-write

The SCU is a DB2 application program and uses the CALL ATTACH FACILITY (CAF) to connect to DB2. It can be executed as a standard MVS/ESA batch job step or under TSO. When the SCU runs under TSO it uses CAF, not the TSO attach facility.

States of IMS DPROP Systems and Propagation Requests

In a synchronous IMS DPROP system, updates to databases are done simultaneously with system processing. This topic describes the various states you can assign to the IMS DPROP system, the IMS and DB2 databases, and to individual propagation requests, and how activating or deactivating the states affects processing.

State of the IMS DPROP System

The states of a IMS DPROP system is recorded in the IMS DPROP status file, which is an MVS file. Using an MVS file lets the RUP and HUP track the status of the IMS DPROP system, even if DB2 or the IMS DPROP directory is unavailable.

The IMS DPROP system can be either:

Emergency Stopped
All synchronous propagation is stopped and all propagation requests are ignored. You use the ESTOP control statement to set the IMS DPROP system status to emergency stopped.

Setting a IMS DPROP system to emergency stopped should be an exceptional action. Set the emergency stopped state when you have severe synchronous propagation problems, such as a long DB2 outage, and must let application programs that update the database run, even though synchronous propagation is not running. Emergency stopping a IMS DPROP system has the disadvantage that you eventually need to resynchronize your IMS and DB2 data.

Not Emergency Stopped
If the IMS DPROP system status is not emergency stopped, IMS DPROP completes synchronous propagation based on the status of individual propagation requests. You use the RESET control statement to set the IMS

DPROP system to not emergency stopped. Unless you specify the DUBIOUS keyword on the RESET statement, RESET deactivates all propagation requests.

States of Individual Propagation Requests

The synchronous processing status of individual propagation requests is recorded in the IMS DPROP directory and in appropriate VLF objects. The status of an individual propagation request can be only one of the following states:

Inactive
When a propagation request is inactive, no synchronous propagation occurs. When first created by the MVG, a propagation request is inactive. After a propagation request is active, you can change the status back to inactive with wither the DEACTIVATE or EDEACTIVATE control statements.

Active When a propagation request is active, synchronous propagation occurs as defined in the propagation request. This is the normal operating status of a propagation request. Use the ACTIVATE control statement to change the status of a propagation request to active.

Suspended
When a propagation request is suspended, no synchronous propagation occurs for that particular request. Suspending propagation requests can be useful when you are running a few explicitly identified, performance-critical IMS batch or BMP jobs that do many updates and cannot tolerate the increase in elapsed time needed to propagate the updates.

For such performance-critical applications, you can:

- Execute the jobs without propagating their updates.
- Develop new programs that apply the same updates to another copy of the data. Then, to reduce elapsed run times, you can execute the DB2 and IMS update applications in parallel.

The concept of suspending propagation requests is different from the concept of deactivating them.

- If you deactivate a propagation request, no updates by *any* application are propagated.
- If you suspend a propagation request, the updates of only *a few, explicitly designated* jobs are not propagated.

  IMS DPROP provides a protection mechanism to prevent propagating jobs from encountering synchronous propagation failures due to data that is temporarily inconsistent. You provide a PROP SUSP control statement in the //EKYIN DD statement of the job step for which you do not want propagation to occur. IMS DPROP does not execute PROP SUSP jobs concurrently with jobs that do not have PROP SUSP control statements.

  The RUP and HUP do not concurrently execute jobs with PROP SUSP control statements and those that do not have PROP SUSP control statements. The RUP and HUP verify that the status of all propagation requests for the segment type or table are compatible with the //EKYIN control statements. If the propagation request is in suspended state, the RUP and HUP allow only updates that have the appropriate PROP SUSP or PROP OFF control statement. If the propagation request is in active state, the RUP and HUP prevent any updates with PROP SUSP //EKYIN control statements.

To avoid mismatched data and eventual propagation failures, updates to the second data copy should be compatible with the mapping logic of IMS DPROP. If you intend to use the CCU to verify data consistency, updates done to the second copy should be identical (to the byte level of fields) to the updates resulting from IMS DPROP mapping logic.

IMS DPROP Support for Orderly and Controlled Status Changes

To preserve consistency between the IMS and DB2 databases, and within the copies of the data being propagated, only change the propagation request status when the databases are not being updated. Performing a propagation request status change when the propagated data is not being updated is called an orderly and controlled status change.

The SCU supports orderly and controlled status changes for one-way IMS-to-DB2, one-way DB2-to-IMS, and two-way propagation for the following control statements:

- ACTIVATE
- DEACTIVATE
- SUSPEND

One-Way IMS-to-DB2 Propagation

For propagation requests defined as MAPDIR=HR, IMS DPROP support for orderly and controlled status changes is based on IMS DBRC share control. DBRC share control ensures that no IMS system has update authority for the affected database when the SCU uses either the ACTIVATE, DEACTIVATE or SUSPEND control word to change the status of the database. Support for orderly and controlled status changes is provided for registered databases only if DBRC share control is active.

Before performing a status change, the SCU ensures that:

- All online IMS subsystems have released their database update authorization through an operator-initiated IMS command.

  Use /DBDUMP or /DBRECOVERY to release the update authorization of an IMS Online system.

  Use the /DBRECOVERY command to release the update authorization of an IMS Online system, for registered Fast Path DEDBs when DBRC share control is active.
- IMS batch subsystems have completed processing the updates to the databases.

  If only IMS batch regions are running, you can use the DBRC CHANGE.DB command, with the keywords READON or NOAUTH, to prevent new updating IMS batch regions from starting.

While waiting for IMS subsystems to release their database authorizations, the SCU lists the conflicting authorizations on OS/VS consoles. It gives one line for each combination of conflicting IMS/ESA subsystem ID and DBD name. The SCU checks every 15 seconds to determine if the conflicting authorizations have been released.

If the SCU must wait, it issues a write-to-operator-with-reply (WTOR), allowing the console operator to reply with either:

LIST To get an up-to-date list of conflicting authorizations

TERM To terminate processing of the SCU. The SCU terminates the current control statement and does not process subsequent control statements in the //SCUIN input file.

If DBRC share control is not active, or if the databases are not DBRC registered, the SCU changes the status without checking or waiting until all IMS subsystems have released their update authorizations and completes with a return code of 4.

It is your responsibility to execute the ACTIVATE, DEACTIVATE, and SUSPEND control statements only when the affected IMS databases are not being updated. Doing so helps to avoid inconsistencies between the IMS and DB2 data, and internally, within the DB2 copy.

One-way DB2-to-IMS Propagation

For propagation requests defined with MAPDIR=RH, the SCU ensures that the status changes requested by ACTIVATE, DEACTIVATE, and SUSPEND control statements are controlled changes. The SCU ensures that these status changes are performed when no DB2 connection has update authorization for the affected DB2 table spaces.

Before performing the requested status changes, the SCU internally issues the following command for the DB2 table spaces of the affected tables:

```
DISPLAY DATABASE(...) SPACENAM(...)
```

The SCU does not perform status changes until all active DB2 connections have released their update authority. While waiting for DB2 connections to release database authority, the SCU lists the conflicting DB2 connections on OS/VS consoles. Then it checks every 15 seconds to determine if the conflicting authorizations have been released.

To prevent new DB2 connections from obtaining update authorizations while the SCU is waiting, you can issue one of the following DB2 commands to set DB2 table spaces and databases to read-only mode:

```
START DATABASE(database-name) SPACENAM(tablespace-name) ACCESS(RO)
START DATABASE(database-name) ACCESS(RO)
```

As with IMS-to-DB2 propagation requests, if the SCU must wait, it issues a WTOR, allowing the console operator to reply with either LIST or TERM.

Two-Way Propagation

For propagation requests defined with MAPDIR=TW, IMS DPROP support for orderly status changes is provided only for registered IMS databases, and if DBRC share control is in effect.

Changes requested by ACTIVATE, DEACTIVATE and SUSPEND control statements are performed when:

- No IMS subsystem has update authority for the affected IMS database.
- No DB2 connection has update authority for the affected DB2 table spaces.

Controlled and orderly status changes are handled the same way as described for one-way propagation.

Emergency Deactivation of Propagation Requests

If severe propagation problems occur, you can emergency deactivate selected propagation requests by invoking the SCU and specifying the EDEACTIVATE

control statement. The SCU deactivates the propagation requests without waiting until updating IMS subsystems and DB2 connections complete their updates and release their update authority.

Applying Propagation Request Status Changes to Groups of Data

Using SCU control statements, you can apply a status change to:

- An individual propagation request
- All propagation requests propagating from or to specified segment types
- All propagation requests propagating from or to specified DBDs
- All propagation requests belonging to propagation request sets

Follow these recommendations when you change the status of a propagation request:

- If you are performing DB2-to-IMS propagation, all propagation requests propagating to the same group of logically related IMS databases should have the same status to avoid propagation failures.
- If you are performing IMS-to-DB2 propagation and you have implemented DB2 RIRs between propagated DB2 tables, all propagation requests to related DB2 tables should have the same status. Doing so helps to avoid propagation failures.

  For example, if a propagation request affecting Table A is altered, then all propagation requests for other tables related to Table A should be altered also.

Refer to the appropriate Administrators Guide for your propagation mode for more information on applying status changes to groups of data.

The SCU considers all propagation request status changes requested by a single control statement as one operation; it performs either none or all of the changes requested on the control statements. Therefore, if you need to apply status changes to a group of propagation requests, request all status changes for a single SCU control statement. This helps ensure that propagation requests for related DB2 tables have the same status.

PROP OFF Mode for Data Repair Programs

PROP OFF stops synchronous propagation of IMS and SQL updates to databases for specific jobs, such as the CCU processing that generates repair statements. Such database repair jobs should update only one copy of the data. When you use PROP OFF, you can run repair programs concurrently with normal operations without impacting normal updates and synchronous propagation. It does not affect the synchronous propagation status of any propagation requests.

PROP OFF mode repair programs contain a PROP OFF control statement in the //EKYIN DD statement of their IMS batch/dependent region JCL. You must authorize PROP OFF mode by running the SCU with the ALLOWPROPOFF control statement.

Execution of database repair programs that use PROP OFF mode should not be confused with either execution of performance-critical programs in PROP SUSP mode or with deactivation of synchronous propagation. It does not change synchronous propagation status from active to inactive or from active to suspended.

Read-Only Status of IMS Databases

When doing synchronous propagation or two-way synchronous propagation, set IMS databases to read-only mode when doing an IMS extract and DB2 load.

Read-only mode ensures that the IMS database is not updated. Read-only also prevents inconsistencies between the IMS database and the DB2 tables being loaded.

Full-Function IMS Databases

If you use DBRC share control and the database is registered in DBRC, you can use the SCU to set the database to read-only mode. All IMS systems must release database update authority. When the SCU gives a return code of zero, you can assume it has set the database to read-only mode and that no IMS systems are updating the database. Then you can begin the extract and load process in the next job step or subsequent job step.

After you complete the extract and load process and runthe associated DB2 utilities (RUNSTATS and COPY), use the SCU READOFF control statement to allow updates to the database.

The sequence of events for performing an extract and load for a full function IMS database is:

1. Execute SCU specifying the READON control statement. This sets the database to read-only status in the DBRC RECON data sets.
2. Do the extract and load and execute the associated DB2 utilities, such as RUNSTATS and COPY.
3. Activate any propagation requests that need to be activated.
4. Execute the SCU specifying the READOFF control statement to make the database available for updates again.

DEDBs

If DBRC share control is not implemented or the database is not registered in DBRC, you cannot use the SCU to mark a full-function database as read-only.

IMS has no concept of read-only status. Neither IMS nor IMS DPROP gives you an automated way of ensuring the DEDB is not updated during the extract and load process. You must devise your own method of ensuring the DEDB is not updated during extract and load. Read-only status is recorded in the IMS RECON data sets, not the IMS DPROP directory or status file.

With two-way synchronous propagation consider setting the propagated IMS databases in read-only access mode while extracting data with, or for, the DLU.

Read-Only Status of DB2 Tables

To prevent inconsistencies between DB2 tables and the IMS database, set the DB2 tables to read-only mode. To set the DB2 tables to read-only mode and perform a DB2 extract and IMS load, use the following steps:

1. Before you start the extract and load, run the SCU specifying the READON control statement. This sets the DB2 table spaces or databases in DB2 to read-only access mode. Internally, the SCU issues a DB2 -START DATABASE ACCESS(RO) command and waits until all active DB2 connections release update authority. When the SCU returns with a return code of zero, the DB2 databases and table spaces are in read-only mode and they are not being updated.
2. Do the DB2 extract and IMS load, and run associated IMS utilities, such as Prefix Resolution/Update and Image Copy. If you are using the DLU, it checks that the DB2 table spaces or databases are in read-only mode and that no concurrent DB2 connection can update the tables being extracted.

3. Activate any propagation requests that need to be propagated.
4. Execute the SCU specifying the READOFF control statement. This sets the DB2 table spaces and databases in DB2 read-write access mode so that DB2 tables can be updated and updates are propagated.

The access mode of DB2 table spaces and databases is recorded in DB2, not in the IMS DPROP directory or status file.

With two-way synchronous propagation, consider setting the propagated DB2 table spaces in read-only mode while extracting the IMS data with Data Refresher or with a user-written program.

Setting and Resetting the Read-Only Status of IMS Databases

IMS DPROP and the Status Change Utility provide protection against IMS updates during the extract and DB2 load only for registered, full-function databases if share control is in effect. If share control is in effect, you can use the Status Change Utility to set the following two database states for registered, full-function IMS databases: read-only (READON) or read-only-off (READOFF).

IMS records these states in the DBRC RECON data sets.

If you call the Status Change Utility to set a registered, full-function database to read-only, the SCU calls the IMS DBRC utility with a CHANGE.DB READON command. The SCU then waits until all IMS subsystems release their database authorization. Therefore, when the SCU returns with a 0 return code, you can assume that the database is not being updated.

If you call the SCU with a READOFF control statement, the SCU calls the IMS DBRC utility with a CHANGE.DB READOFF command.

Setting and Resetting the Read-Only Status of DB2 Table Spaces and Databases

IMS DPROP and the SCU provide protection against DB2 updates during the extract and IMS load of IMS databases.

You can call the SCU with a READON control statement to set the read-only access mode of DB2 table spaces and DB2 databases. When calling the SCU with a READON control statement, the SCU issues one of the following DB2 commands:

```
START DATABASE(database-name) SPACENAM(tablespace-name) ACCESS(RO)
START DATABASE(database-name) ACCESS(RO)
```

The SCU then waits until the table space/database has been set to read-only and all DB2 connections release their update authorization. Therefore, when the SCU returns with a zero return code, you can assume that the table space/database is not being updated.

If you call the SCU with a READOFF control statement to reset the read-only state, the SCU issues one of the following DB2 commands:

```
START DATABASE(database-name) SPACENAM(tablespace-name) ACCESS(RW)
START DATABASE(database-name) ACCESS(RW)
```

Control statements for synchronous propagation

This topic describes the control statements associated with the SCU for synchronous propagation.

The Propagation Request Identification Block

Many of the SCU control statements have the same combination of keywords. The combination of the PR, PRSET, DBD, SEG, and ALL keywords comprise a “PR identification block” that you will see repeated in many of the SCU control statements. Use the propagation request identification block to describe the propagation requests to which the SCU control statement should be applied.

Unless noted otherwise, the syntax of the SCU control statements is as described in Figure 68.

```
►►─STATEMENT NAME─┤ PR Identification Block ├─;──────────────────────────►◄
```

PR Identification Block:

```
                  ┌─,──────┐
├─┬─PR=(──────────┴─pr_id──┴─)──────────────────────────────────────────┬─┤
  │                 ┌─,──────┐                                          │
  ├─PRSET=(─────────┴─prset_id┴─)──────────────────────────────────────┤
  │                 ┌─,──────┐                                          │
  ├─DBD=(───────────┴─dbd_name┴─)──┬──────────────────────────────────┤
  │                                │                ┌─,──────┐         │
  │                                └─,PRSET=(───────┴─prset_id┴─)──────┘
  │                 ┌─,──────┐                                          │
  ├─DBD=dbd_name,SEG=(──────┴─seg_name┴─)──┬──────────────────────────┤
  │                                        │                ┌─,──────┐ │
  │                                        └─,PRSET=(───────┴─prset_id┴─)─┘
  └─ALL──────────────────────────────────────────────────────────────────┘
```

Figure 68. Example of an SCU control statement

PR=(*prid1,prid2,...* **)**
Identifies a propagation request or a list of propagation requests.

PRSET=(*prset1,prset2,...* **)**
Identifies a PRSET or a list of PRSETs. By using this keyword alone or together

with a DBD keyword, you can limit the scope of the control statement to the propagation requests belonging to the specified PRSETs.

DBD=(*database1,database2,...* **)**
Identifies a physical IMS database, or a list of physical IMS databases.

When you use this keyword, the SCU control statement is applied to:
- All propagation requests from the specified IMS databases for IMS-to-DB2 propagation
- All propagation requests to the specified IMS databases for DB2-to-IMS propagation

DBD= *database* **,SEG=(***segname1,segname2,...***)**
Identifies a physical IMS segment type, or a list of IMS segment types.

You must specify one physical IMS database name on the DBD keyword when using the SEG keyword. When you specify the combination of DBD= and SEG= keywords, the SCU control statement is applied to all propagation requests from and to the specified IMS segment types.

Do not specify the name of internal segments in the SEG= keyword. Internal segments are not IMS segments; instead they represent structures that are embedded within a containing IMS segment, and that are propagated by mapping case 3 propagation requests.

When requesting an SCU function for propagation requests propagating internal segments, you can specify either:
- The PR IDs on the PR= keyword
- The name of the containing IMS segment in the SEG= keyword

If you specify the name of an IMS segment containing internal segments in the SEG= keyword, then the SCU control statement is applied to:
- All propagation requests propagating the internal segments
- All propagation requests propagating the containing IMS segment

If an IMS segment specified in the SEG= keyword contains fields used as path data, then:
- The SCU control statement is applied *only* to those propagation requests propagating the identified segment as an entity or extension segment (and to those propagation requests for internal segments contained in the identified IMS segment).
- The SCU control statement is *not* applied to those propagation requests for dependent segments and including path data from the identified segment.

ALL
Specifies all propagation requests for a IMS DPROP system.

Note: If you use the ALL keyword with the ACTIVATE, DEACTIVATE, or SUSPEND control statements, all the propagated IMS databases and DB2 tables are affected.

For IMS-to-DB2 propagation, if DBRC share control is in effect and if the databases are registered, IMS DPROP waits until all of the propagated IMS databases have their update authorization released before processing the SCU control statement. For DB2-to-IMS propagation, IMS DPROP waits until all of the propagated DB2 table spaces have their update authorization released before processing the SCU control statement. In a production environment this is not always convenient. Use caution, therefore, when using the ALL keyword with these control statements.

The control statements ACTIVATE, DEACTIVATE, SUSPEND, TRACEON, TRACEOFF, ERROPT apply to individual propagation requests. However, it is recommended that you code them *with* the DBD and SEG keywords so that the change is applied to all propagation requests for the identified segments or DBDs.

ACTIVATE

Use the ACTIVATE statement to activate a propagation request in an orderly and controlled way. You can also use the abbreviation ACT. You can use this statement to:

- Activate a propagation request after the IMS extract and DB2 load is completed.
- Reactivate propagation of a previously deactivated propagation request.
- Reactivate propagation of a previously suspended propagation request.

For IMS-to-DB2 propagation, IMS DPROP support for orderly and controlled status changes is based on IMS DBRC share control. To ensure that your status changes are orderly, your databases must be registered, and DBRC share control must be in effect. For more information on DBRC share control, refer to IMS/ESA Operations Guide.

Syntax of the ACTIVATE control statement

Figure 69 shows the syntax of the ACTIVATE statement. For a description of the propagation request identification block, refer to “The Propagation Request Identification Block ” on page 188.

```
>>──ACTIVATE──────────────────────────────────────────────────────────>
>──┬─PR=(──┬─pr_id──┬──)──────────────────────────────────────────────┬──;──────>>
  │        └──,──────┘                                                │
  ├─PRSET=(──┬─prset_id──┬──)──────────────────────────────────────────┤
  │          └──,────────┘                                            │
  ├─DBD=(──┬─dbd_name──┬──)──┬──────────────────────────────────────┤
  │        └──,──────────┘    │                                      │
  │                           └─,PRSET=(──┬─prset_id──┬──)──┘        │
  ├─DBD=dbd_name,SEG=(──┬─seg_name──┬──)──┬──────────────────────┤
  │                     └──,────────┘     │                        │
  │                                       └─,PRSET=(──┬─prset_id──┬──)──┘
  └─ALL──────────────────────────────────────────────────────────────┘
```

Figure 69. Syntax of the ACTIVATE control statement

Examples of the ACTIVATE Statement: The example in Figure 70 on page 191 show how to use the ACTIVATE control statement.

```
ACT PR=(PR103,PR105)
ACT DBD=(SKILLDB,INVENTRY),PRSET=(SET1,SET5);
```

Figure 70. Example of ACTIVATE

ALLOWPROPOFF

Use the ALLOWPROPOFF statement to allow execution of database repair job steps having PROP OFF input control statements in the //EKYIN data set. You can also use the abbreviation ALLOW. The syntax of the ALLOWPROPOFF command is shown in Figure 71. For a description of the propagation request identification block, refer to “The Propagation Request Identification Block ” on page 188.

Included in this statement is the optional JOB keyword. If you specify a particular job on the JOB keyword, only that job, with the PROP OFF statement in the //EKYIN data set, is authorized to run. Omitting the JOB keyword ensures that all jobs with the PROP OFF statement are authorized to run.

Syntax of the ALLOWPROPOFF control statement

Figure 71 gives the syntax of ALLOWPROPOFF.

```
►►──ALLOWPROPOFF──────────────────────────────────────────────────────►

►──┬─PR=(──┬─pr_id──┬──)──────────────────────────────────────────────┬──►
   │       └──,──────┘                                                │
   ├─PRSET=(──┬─prset_id──┬──)──────────────────────────────────────────┤
   │          └──,─────────┘                                          │
   ├─DBD=(──┬─dbd_name──┬──)──┬──────────────────────────────────────┤
   │        └──,────────┘     │                                      │
   │                          └─,PRSET=(──┬─prset_id──┬──)──┘         │
   ├─DBD=dbd_name,SEG=(──┬─seg_name──┬──)──┬──────────────────────┤
   │                     └──,──────────┘     │                      │
   │                                         └─,PRSET=(──┬─prset_id──┬──)──┘
   └─ALL──────────────────────────────────────────────────────────────┘

►──┬──────────────────┬──;──────────────────────────────────────────►◄
   └─,JOB=job_name────┘
```

Figure 71. Syntax of the ALLOWPROPOFF control statement

Examples of the ALLOWPROPOFF Statement: Figure 72 shows you how to use the ALLOWPROPOFF control statement.

```
ALLOW DBD=SKILLDB
ALLOW PR=PR103,JOB=T000000;
```

Figure 72. Example of ALLOWPROPOFF

DEACTIVATE

Use the DEACTIVATE command to deactivate propagation for a propagation request in an orderly, controlled way. You can also use the abbreviation DEA.

For IMS-to-DB2 propagation, IMS DPROP support for orderly and controlled status changes is based on IMS DBRC share control. To ensure that your status changes are orderly, your databases must be registered and DBRC share control must be in effect. For more information on DBRC share control, refer to *IMS/ESA Operations Guide*.

Syntax of the DEACTIVATE control statement

Figure 73 shows the syntax of the DEACTIVATE command. For a description of the propagation request identification block, refer to “The Propagation Request Identification Block ” on page 188.

```
►►──DEACTIVATE──────────────────────────────────────────────────►
                                                                 
►──┬─PR=(──┬─pr_id──┬─)──────────────────────────────────────────┬──;──►◄
   │       └──,─────┘                                            │
   │─PRSET=(──┬─prset_id──┬─)──────────────────────────────────── │
   │          └──,────────┘                                      │
   │─DBD=(──┬─dbd_name──┬─)──┬──────────────────────────────────┤
   │        └──,───────┘    │                                   │
   │                        └─,PRSET=(──┬─prset_id──┬─)─────────┤
   │                                    └──,────────┘           │
   │─DBD=dbd_name,SEG=(──┬─seg_name──┬─)──┬──────────────────────┤
   │                     └──,──────────┘    │                   │
   │                                        └─,PRSET=(──┬─prset_id──┬─)─┤
   │                                                    └──,────────┘   │
   └─ALL──────────────────────────────────────────────────────────────┘
```

Figure 73. Syntax of the DEACTIVATE control statement

Examples of the DEACTIVATE Statement: Figure 74 on page 193 shows you how to use the DEACTIVATE control statement.

```
DEA DBD=SKILLDB,SEG=ROOT
DEA PR=PR103;
```

Figure 74. Example of DEACTIVATE

DENYPROPOFF

Use the DENYPROPOFF statement to deny execution of database repair job steps having PROP OFF control statements in the //EKYIN data set. You can also use the abbreviation DENY.

Syntax of the DENYPROPOFF control statement

Figure 75 shows the syntax of the DENYPROPOFF statement. For a description of the propagation request identification block, refer to “The Propagation Request Identification Block ” on page 188.

```
►►──DENYPROPOFF──────────────────────────────────────────────────►

                  ┌─,─────────┐
►──┬─PR=(──────────▼─pr_id────┴─)──────────────────────────────┬─;──────────►◄
   │                ┌─,─────────┐
   ├─PRSET=(────────▼─prset_id──┴─)──────────────────────────────┤
   │                ┌─,─────────┐
   ├─DBD=(──────────▼─dbd_name──┴─)──┬──────────────────────────┤
   │                                 │              ┌─,─────────┐│
   │                                 └─,PRSET=(─────▼─prset_id───┴┘│
   │                ┌─,─────────┐                                │
   ├─DBD=dbd_name,SEG=(─────────▼─seg_name──┴─)──┬──────────────┤
   │                                             │ ┌─,─────────┐│
   │                                             └─,PRSET=(─────▼─prset_id───┴┘│
   └─ALL──────────────────────────────────────────────────────────┘
```

Figure 75. Syntax of the DENYPROPOFF control statement

Examples of the DENYPROPOFF Statement: Figure 76 gives examples of DENYPROPOFF.

```
DENY DBD=SKILLDB
DENY DBD=(SKILLDB,INVENTRY),PRSET=(SET1,SET5);
```

Figure 76. Example of DENYPROPOFF

DISPLAY

Use the DISPLAY statement to show:

- Whether the IMS DPROP system is emergency stopped
- The total amount of virtual storage required by the VLF objects of this IMS DPROP system

The system administrator can use the information on virtual storage to adjust the MAXVIRT value for the VLF class in the COFVLFxx member. For more information on the MAXVIRT value, refer to*IMS DPROP Installation Guide*.

Figure 77 shows the syntax of the DISPLAY statement.

```
►►──DISPLAY──STATUS──;──────────────────────────────────────────────►◄
```

Figure 77. Syntax of the DISPLAY control statement

You must provide the keyword STATUS on the DISPLAY statement.

Example of the DISPLAY Statement

Figure 78 is an example of how to display IMS DPROP system status.

```
DISPLAY STATUS
```

Figure 78. Example of DISPLAY control statement

Example of a DISPLAY Response

Figure 79 is an example of the response that results from a DISPLAY command.

```
EKYS467I DPROP SYSTEM STATUS: PROPAGATION IS ACTIVE
         DPROP SYSTEM NAME: T003815
         DPROP SYSTEM LEVEL: 110

EKYS468I VLF OBJECT SIZE: 44928 BYTES
         VLF CLASS NAME: PM3
```

Figure 79. Example Response to a DISPLAY control statement

EDEACTIVATE

Use the EDEACTIVATE statement for an emergency deactivation of a subset of the propagation requests. You can also use the abbreviation EDE. When you ask for an emergency deactivation, it can take a few seconds for the RUP and HUP to be notified. Also, the subsequent deactivation in the various propagating regions is not simultaneous.

Syntax of the EDEACTIVATE control statement

Figure 80 on page 195 shows the syntax of the EDEACTIVATE statement. For a description of the propagation request identification block, refer to “The Propagation Request Identification Block ” on page 188.

```
►►──EDEACTIVATE──────────────────────────────────────────────────────►
                                                                      
                  ┌─,─┐                                               
►──┬─PR=(──┬─pr_id──┴─)──────────────────────────────────────────┬─;──►◄
   │       │                                                     │
   ├─PRSET=(──┬─prset_id──┴─)──────────────────────────────────────┤
   │          │                                                  │
   ├─DBD=(──┬─dbd_name──┴─)──┬──────────────────────────────────┤
   │        │                │                                  │
   │        │                └─,PRSET=(──┬─prset_id──┴─)──────┘
   │        │                            │                      
   ├─DBD=dbd_name,SEG=(──┬─seg_name──┴─)──┬──────────────────┤
   │                     │                │                  │
   │                     │                └─,PRSET=(──┬─prset_id──┴─)──┘
   │                     │                            │                 
   └─ALL──────────────────────────────────────────────────────────┘
```

Figure 80. Syntax of the EDEACTIVATE control statement

Examples of the EDEACTIVATE Statement: Figure 81 shows how to use the EDEACTIVATE Control statement.

```
EDE DBD=SKILLDB,PRSET=SET1
EDE PR=PR103;
```

Figure 81. Example of EDEACTIVATE

ERRCTL

Use the ERRCTL statement to change control information in the IMS DPROP directory. The error-reporting logic of the RUP and HUP use this control information to protect against flooding the OS/VS consoles and the audit trail with error messages generated while encountering propagation failures with ERROPT=IGNORE.

The RUP and HUP count the number of propagation failures that do not result in a backout (for example, propagation failures of a propagation request with ERROPT=IGNORE that are not caused by unavailable resources). During one monitoring period of 15 minutes, the RUP and HUP count these errors for both individual propagation requests and for the whole IMS DPROP system. The propagation request table of the IMS DPROP directory contains the value for the maximum number of failures associated with a propagation request that should be reported within a 15-minute interval to the OS/VS consoles and to the audit trail. The master table of the IMS DPROP directory contains the value for the maximum number of failures associated with the IMS DPROP system that should be reported. You can set these maximum numbers during propagation request generation or with the SCU.

If the number of errors exceeds the value you specified, the RUP and HUP suppress the error messages to the OS/VS consoles and the audit trail (but still writes error messages to the IMS DPROP trace output and to the optional //EKYPRINT data set).

The SCU allows you to reset the error counts to zero and start a new error-reporting period. Then the RUP and HUP resume writing error messages to the OS/VS consoles and the audit trail. You might want to start a new error-reporting period after fixing a propagation problem so you can immediately see if the RUP and HUP are still encountering the same problem.

Syntax of the ERRCTL control statement

Figure 82 shows the syntax of the ERRCTL statement. For a description of the propagation request identification block, refer to “The Propagation Request Identification Block ” on page 188.

```
                  ,
                  ┌──────────────────────────────────────────────────┐
►►──ERRCTL──┬──┬──┴──MAXPR=──┬─integer─┬──,──┤ PR Identification Block ├──┬──;──►◄
            │  │             └─UNLIMITED─┘                                │
            │  ├──MAXSSWTO=──┬─integer─┬──────────────────────────────────┤
            │  │             └─UNLIMITED─┘                                │
            │  ├──MAXSSAUD=──┬─integer─┬──────────────────────────────────┤
            │  │             └─UNLIMITED─┘                                │
            │  └──NEWCYCLE──────────────────────────────────────────────┘
```

PR Identification Block:

```
                  ,
                  ┌──────┐
├──┬──PR=(──┬──pr_id──┬──)──────────────────────────────────────────────────┤
   │                  ,
   │                  ┌──────┐
   ├──PRSET=(──┬──prset_id──┬──)──────────────────────────────────────────┤
   │                  ,
   │                  ┌──────┐
   ├──DBD=(──┬──dbd_name──┬──)──┬──────────────────────────────────────┤
   │                  ,
   │                  ┌──────┐
   │                  └──,PRSET=(──┬──prset_id──┬──)──┘
   │                  ,
   │                  ┌──────┐
   ├──DBD=dbd_name,SEG=(──┬──seg_name──┬──)──┬──────────────────────────┤
   │                  ,
   │                  ┌──────┐
   │                  └──,PRSET=(──┬──prset_id──┬──)──┘
   └──ALL──────────────────────────────────────────────────────────────────┘
```

Figure 82. Syntax of the ERRCTL control statement

MAXPR=integer] UNLIMITED

Limits the maximum number of failures for a PR that are:

- Reported to the OS/VS consoles and audit trail in one monitoring cycle
- Documented with detailed trace records in the IMS DPROP trace in one monitoring cycle

Use any positive value between 0 and 2,147,483,647.

You can set this keyword to zero if a propagation request generates situations that the RUP and HUP consider propagation failures. Setting MAXPR=0 prevents error messages from being written to the OS/VS consoles and audit trail.

MAXSSWTO=integer] UNLIMITED
Limits the maximum number of failures for the IMS DPROP system that are:

- Reported on the OS/VS consoles in one monitoring cycle
- Documented with detailed trace records in the IMS DPROP trace in one monitoring cycle

Use any positive value between 0 and 2,147,483,647.

MAXSSAUD=integer] UNLIMITED
Limits the maximum number of failures for the IMS DPROP system reported on the audit trail in one monitoring cycle. You can use any positive value between 0 and 2,147,483,647.

NEWCYCLE
Starts a new monitoring cycle. For example, you would select NEWCYCLE when you want the RUP and HUP to resume writing error messages.

It can take a few seconds for the RUP and HUP to be notified that the control information has been changed. Also, the various propagating regions are not notified of this change simultaneously.

Examples of the ERRCTL Statement

Figure 83 gives some examples of how to use the ERRCTL control statement.

```
ERRCTL MAXPR=0,PR=(PR103,PR105)
ERRCTL NEWCYCLE;
ERRCTL MAXSSAUD=100,MAXSSWTO=30,NEWCYCLE;
```

Figure 83. Examples of ERRCTL

ERROPT

Use the ERROPT statement to change the error option, IGNORE or BACKOUT, of a propagation request.

It can take a few seconds for the RUP and HUP to be notified that the error option has been changed. Also, the various propagating regions are not notified of this change simultaneously.

Syntax of the ERROPT control statement

Figure 84 on page 198 shows the syntax of the ERROPT statement. For a description of the propagation request identification block, refer to “The Propagation Request Identification Block ” on page 188.

```
►►─ERROPT──┬─IGNORE─┬──────────────────────────────────────────────►
           └─BACKOUT─┘

►──┬─,PR=(──┬─pr_id──┬─)──────────────────────────────────────────;──►◄
   │        ┌──,──┐
   ├─,PRSET=(──┬─prset_id──┬─)──────────────────────────────────────┤
   │        ┌──,──┐
   ├─,DBD=(──┬─dbd_ name──┬─)──┬──────────────────────────────────┤
   │                           │        ┌──,──┐                    │
   │                           └─,PRSET=(──┬─prset_id──┬─)─────────┘
   │                  ┌──,──┐
   ├─,DBD=dbd_name,SEG=(──┬─seg_name──┬─)──┬──────────────────────┤
   │                                        │        ┌──,──┐        │
   │                                        └─,PRSET=(──┬─prset_id──┬─)─┘
   └─,ALL──────────────────────────────────────────────────────────┘
```

Figure 84. Syntax of the ERROPT control statement

IGNORE] BACKOUT
To ignore errors issued when propagation cannot be successfully completed, select IGNORE. To back out the changes made since the last commit point when errors occur, select BACKOUT.

Examples of the ERROPT Statement

Figure 85 shows how to use the ERROPT control statement.

```
ERROPT IGNORE,DBD=SKILLS
ERROPT BACKOUT,DBD=SKILLS,SEG=ROOT;
```

Figure 85. Examples of ERROPT

ESTOP

Use the ESTOP statement to perform an emergency stop of all propagating activities of the entire IMS DPROP system. As with the EDEACTIVATE statement, it can take a few seconds for the RUP and HUP to be notified of the emergency stop. Also, the subsequent stopping in the various propagating regions is not simultaneous.

Syntax of the ESTOP control statement

Figure 86 on page 199 shows the syntax of the ESTOP statement.

```
►►──ESTOP──DPROP──;──────────────────────────────────────────────────►◄
```

Figure 86. Syntax of the ESTOP control statement

Example of the ESTOP Statement: Figure 87 shows how to use the ESTOP control statement.

```
ESTOP DPROP
```

Figure 87. Example of ESTOP

INIT

Use the INIT statement to initialize IMS DPROP control information.

INIT is one of two SCU statements supported in a user asynchronous IMS DPROP system. The other is DISPLAY STATUS.

Syntax of the INIT control statement

Figure 88 shows the syntax of the INIT statement.

```
►►──INIT──┬─DPROP=dpropname──────────┬──;──────────────────────────►◄
          ├─STATF──┬──────────────┬───┤
          │        └─,ESTOPPED────┘   │
          └─VLF───────────────────────┘
```

Figure 88. Syntax of the INIT control statement

DPROP= *dpropname*

Initializes the IMS DPROP directory and the status file of the IMS DPROP system you identify. Define *dpropname* at DPROPGEN time.

When you issue the INIT DPROP statement, the SCU verifies that the master table in the IMS DPROP directory is empty. If it is not empty, the SCU writes an error message and stops further processing.

If the SCU finds the directory empty, it inserts the unique row of the master table of the IMS DPROP directory. This row contains the IMS DPROP system name that the system administrator provided during IMS DPROP customization and a unique token generated by the SCU. At this time, the SCU also creates the unique record of the status file.

For more information on the master table, refer to Chapter 1, “Common JCL for IMS DPROP Components,” on page 3.

STATF

Re-initializes or rebuilds a status file. For example, you would issue this keyword if the status file is damaged or lost.

This keyword is not supported in a user asynchronous IMS DPROP system.

If you want the rebuilt status file to indicate that the IMS DPROP system is emergency stopped, issue the control statement with the ESTOPPED keyword, INIT STATF, ESTOPPED.

VLF

Builds all VLF objects, which contain IMS DPROP control information. All objects of the VLF class belonging to the IMS DPROP system are deleted before being recreated.

You might use this keyword to populate VLF after an IPL or after a sequence of VLF deactivation and reactivation. You should use an INIT VLF statement to populate VLF rather than let the RUP and HUP populate VLF on a demand basis. This avoids some ENQueue conflicts on the small IMS DPROP directory tables.

You can also use an INIT VLF after a software error created wrong objects in VLF, or after recovering the IMS DPROP directory tables.

This keyword is not supported in a user asynchronous IMS DPROP system.

Examples of the INIT Statement

Figure 89 shows examples of the INIT control statement.

```
INIT STATF,ESTOPPED
INIT DPROP=TEST01;
```

Figure 89. Example of INIT

READOFF

Use the READOFF statement to:

- Reset the DBRC read-only mode of an IMS full-function database.

  The READOFF statement supports full-function IMS databases only if DBRC share control is in effect and the database is registered in DBRC.
- Reset the DB2 read-only access mode (set the DB2 read-write access mode) of a DB2 database or table space.

For example, the database administrator might call the SCU with a READOFF statement after completing an extract and load of propagated data. This allows subsequent updates to be made to the propagated data.

Before issuing the READOFF statement, you should activate the affected propagation requests by issuing the ACTIVATE statement. Activating propagation requests prevents the data from being updated without being propagated and also prevents the loss of synchronization between the IMS and DB2 data copies.

Syntax of the READOFF control statement

Figure 90 shows the syntax of READOFF for full-function IMS databases,

```
                  ┌─,──────────┐
►►──READOFF──DBD=(──▼──dbd_name──)──;──────────────────────────────►◄
```

Figure 90. Syntax of the READOFF Statement for Full-Function IMS Databases

DBD=(*database1,database2,...* **)**

Resets the read-only mode of a physical IMS database or a list of physical IMS databases.

Figure 91 shows the syntax of READOFF for DB2.

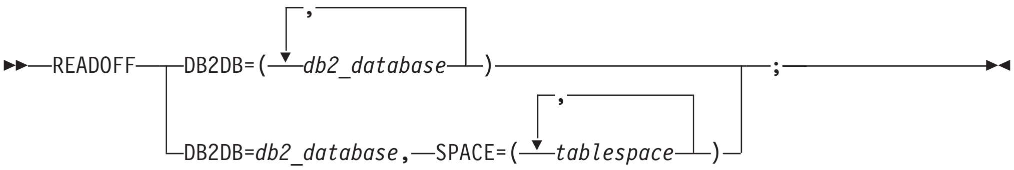

Figure 91. Syntax of the READOFF Statement for DB2 Databases and DB2 Table Spaces

DB2DB=(*db2-database1,db2-database2,...***)**
Resets the read-only access mode (sets the read-write access mode) of a DB2 database or a list of DB2 databases.

DB2DB= *db2-database* **,SPACE=(***tablespace1,tablespace2,...***)**
Sets the read-only access mode (sets the read-write access mode) of a DB2 table space or a list of DB2 table spaces located in the same DB2 database.

Examples of the READOFF Statement for IMS and DB2

The first example in Figure 92 shows an example of READOFF for IMS, the next two examples in Figure 92 show examples for DB2.

```
READOFF DBD=SKILLDB
READOFF DB2DB=(SALARY,ACCOUNT);
READOFF DB2DB=ACCOUNT,SPACE=HISTORY;
```

Figure 92. Examples of READOFF for IMS and DB2

READON

Use the READON statement to:

- Set an IMS full-function database in DBRC read-only mode.

  The READON statement supports full-function databases only if DBRC share control is in effect and if the database is registered in DBRC.
- Set DB2 table spaces or DB2 databases in DB2 read-only access mode.

The database administrator, for example, would call the SCU with the READON statement before starting an extract and load of propagated data. This prevents updates to the propagated data during the extract and load process.

When processing a READON control statement identifying IMS databases, the SCU waits until updating IMS subsystems have released their database update authority. After successful completion of the SCU, you can assume that the IMS database cannot be updated. However, if you directly call the DBRC utility to process CHANGE.DB READON commands, DBRC does not wait until updating IMS systems release their update authority before returning. For more information on DBRC share control, refer to IMS/ESA Operations Guide.

When processing a READON control statement identifying DB2 databases or DB2 table spaces, the SCU waits until updating DB2 connections have released their

table space update authority. After successful completion of the SCU, you can assume that the DB2 table space or DB2 database cannot be updated. However, if you directly issue the following command:

```
START DATABASE(....)   SPACENAM(....)   ACCESS(RO)
```

DB2 does not wait until updating DB2 connections release their update authority before returning. For more information on this command, refer to DB2 Command Reference.

Syntax of the READON control statement

Figure 93 shows the syntax of READON for full-function IMS databases, Figure 94 shows the syntax of READON for DB2.

```
                  ┌─,──────────┐
                  ▼            │
►►─READON─DBD=(──dbd_name──────┴─)─;──────────────────────────────────►◄
```

Figure 93. Syntax of the READON Statement for Full-Function IMS Databases

DBD=(*database1,database2,...* **)**
Sets the read-only mode of a physical IMS database or a list of physical IMS databases.

```
                  ┌─,──────────┐
                  ▼            │
►►─READON──┬─DB2DB=(──db2_database──────┴─)──────────────────────┬─;──────────►◄
           │                                                     │
           │                          ┌─,──────────┐             │
           │                          ▼            │             │
           └─DB2DB=db2_database,─SPACE=(──tablespace──────┴─)────┘
```

Figure 94. Syntax of the READON Statement for DB2 Databases and DB2 Table Spaces

DB2DB=(*db2_database1,db2_database2,...***)**
Sets the read-only access mode of a DB2 database or a list of DB2 databases.

DB2DB= *db2_database* **, SPACE=(***tablespace1,tablespace2,...***)**
Sets the read-only access mode of a DB2 table space or a list of DB2 table spaces located in the same DB2 database.

Examples of the READON Statement

The first example in Figure 95 shows READON used for an IMS database. The second and third example in Figure 95 shows READON for DB2.

```
READON DBD=(SKILLDB,INVENTRY)
READON DB2DB=(SALARY,ACCOUNT);
READON DB2DB=ACCOUNT,SPACE=HISTORY;
```

Figure 95. Examples of READON for IMS and DB2

RESET

Use the RESET statement to reset the emergency-stopped status of a IMS DPROP system. You can use the RESET statement in two ways:

- You can reset the emergency-stopped IMS DPROP system in an orderly and controlled way by simply issuing a RESET IMS DPROP control statement. During an orderly reset, the SCU marks all propagation requests as inactive in the IMS DPROP directory. Next, it sets a flag in the status file indicating that the IMS DPROP system is active. When you issue a RESET DPROP statement, the installation typically resynchronizes the IMS and DB2 data *before* reactivating the propagation requests.
- Alternatively, you can reset the emergency-stopped IMS DPROP system by issuing a RESET DPROP DUBIOUS statement. During a dubious reset, the SCU sets a flag in the status file indicating that the IMS DPROP system is active, *without* marking all propagation requests as inactive. Data propagation is then reinstated without prior resynchronization of the data. You would use DUBIOUS when you want to assume the risk of inconsistent data, resume propagation, and defer resynchronization of the IMS and DB2 data until a later time.

Attention: Using the RESET DPROP DUBIOUS statement often results in propagation failures due to inconsistencies and is therefore *not recommended*.

If you issue the RESET DPROP DUBIOUS statement, it can take a few seconds for the RUP and HUP to be notified of the reset. Also, the subsequent reset in the various propagating regions is not simultaneous.

Syntax of the RESET control statement

Figure 96 shows the syntax of the RESET statement.

```
►►──RESET──DPROP──┬──────────┬──;──────────────────────────────────────►◄
                  └─,DUBIOUS─┘
```

Figure 96. Syntax of the RESET control statement

Example of the RESET Statement: Figure 97 shows an example of the RESET control statement.

```
RESET DPROP
```

Figure 97. Example of RESET

SUSPEND

Use the SUSPEND statement to temporarily suspend the propagation activities of some propagation requests if you need to execute performance critical, updating batch or BMP programs. You can also use the abbreviation SUS. To reactivate the suspended propagation requests, use the ACTIVATE statement.

When a propagation request is suspended, IMS DPROP:

- Does not perform propagation for the suspended propagation requests

- Makes sure that only job steps having matching PROP SUSP control statements in their //EKYIN file can update the data propagated by the suspended propagation requests

Refer to the appropriate Administrators Guide for your propagation mode for more information on this type of protection.

For IMS-to-DB2 propagation, IMS DPROP support for orderly and controlled status changes is based on IMS DBRC share control. To ensure that your status changes are orderly, your databases must be registered and DBRC share control must be in effect. For more information on DBRC share control, refer to *IMS/ESA Operations Guide*.

Syntax of the SUSPEND control statement

Figure 98 shows the syntax of the SUSPEND statement. For a description of the propagation request identification block, refer to “The Propagation Request Identification Block ” on page 188.

```
►►──SUSPEND──────────────────────────────────────────────────────►
                                  ┌─,──────────┐
►──┬─PR=(──┬─pr_id──┬─)──────────────────────────────────────────┬──;──────►◄
   │       └────────┘                                            │
   ├─PRSET=(──┬─,──────────┐──┬─prset_id──┬─)──────────────────────┤
   │          │            │  └───────────┘                      │
   │          ▼            │                                     │
   ├─DBD=(──┬─dbd_name──┬─)──┬──────────────────────────────────┬──┤
   │        │           │    │                                  │  │
   │        ▼           │    │                ┌─,──────────┐    │  │
   │                    │    │                │            │    │  │
   │                    │    └─,PRSET=(──┬─prset_id──┬─)──┘    │  │
   │                    │                      └───────────┘      │  │
   │                    │                                         │  │
   │                    │                ┌─,──────────┐           │  │
   ├─DBD=dbd_name,SEG=(──┬─seg_name──┬─)──┬──────────────────────┬──┤
   │                    │           │    │                      │  │
   │                    ▼           │    │                ┌─,──────────┐
   │                                │    │                │            │
   │                                │    └─,PRSET=(──┬─,──┬─prset_id──┬─)──┘
   │                                │                    │   └───────────┘
   │                                │                    ▼
   └─ALL──────────────────────────────────────────────────────────┘
```

Figure 98. Syntax of the SUSPEND control statement

Examples of the SUSPEND Statement: Figure 99 shows you how to use the SUSPEND control statement.

```
SUS PR=PR103
SUS DBD=SKILLDB,SEG=(ROOT,CHILD1);
```

Figure 99. Example of SUSPEND

TRACEOFF

Use the TRACEOFF statement to deactivate tracing previously activated with the TRACEON statement.It can take a few seconds for the RUP and HUP to be notified that the trace has been deactivated. Also, the various propagating regions are not notified of the deactivation simultaneously.

Syntax of the TRACEOFF control statement

Figure 100 shows the syntax of the TRACEOFF control statement.

```
                          ┌─,──────────┐
                          ▼            │
►►─TRACEOFF──┬─DBD=(──dbd_name──)──────────────┬──;──────────────────►◄
             │                                 │
             │                   ┌─,──────┐    │
             │                   ▼        │    │
             ├─DBD=dbd_name,SEG=(──seg_name──)─┤
             └─ALL─────────────────────────────┘
```

Figure 100. Syntax of the TRACEOFF control statement

DBD=(*database1,database2,...* **)**
Deactivates previously activated tracing for a physical IMS database or a list of physical IMS databases.

DBD= *database* **,SEG=(***segment1,segment2,...***)**
Deactivates previously activated tracing for a physical IMS segment type or a list of IMS segment types.

You must specify one physical IMS database name on the DBD= keyword when using the SEG= keyword.

ALL
Deactivates previously activated tracing of all registered propagation requests of a IMS DPROP system.

Examples of the TRACEOFF Statement

Figure 101 shows two examples of how to use the TRACEOFF control statement

```
TRACEOFF DBD=SKILLDB,SEG=(ROOT,CHILD1)
TRACEOFF ALL;
```

Figure 101. Examples of TRACEOFF

TRACEON

Use the TRACEON statement to activate the tracing of propagation activities from and to a specific IMS database or from and to specific segment types. When you issue the TRACEON statement, tracing of the propagation activities of all job steps propagating from and to the specified IMS databases or segment types begins.

You can trace the propagation activities of a specific job step rather than all propagating job steps by issuing a TRACE control statement in the //EKYIN file of that job step. For more information about this type of trace, see “TRACE ” on page 17.

As with other SCU control statements, it can take a few seconds for the RUP and HUP to be notified that a trace has been set. Also, tracing in the various propagating regions does not begin simultaneously.

Syntax of the TRACEON control statement

Figure 102 shows the syntax of the TRACEON statement.

```
                                  ┌─,──────────┐
                                  ▼            │
►►─TRACEON─DEBUG=integer──┬─,DBD=(──dbd_name──┴─)──────────────┬─;──────────►◄
                          │                                    │
                          │                  ┌─,──────────┐    │
                          ├─,DBD=dbd_name,SEG=(──seg_name──┴─)─┤
                          └─,ALL───────────────────────────────┘
```

Figure 102. Syntax of the TRACEON control statement

DBD=(*database1,database2,...* **)**
Activates tracing of a physical IMS database, or a list of physical IMS databases.

DBD= *database* **,SEG=(***segment1,segment2,...***)**
Activates tracing of a physical IMS segment type, or a list of IMS segment types.

You must specify one physical IMS database name on the DBD= keyword when using the SEG= keyword.

ALL
Activates tracing of all propagation requests of a IMS DPROP system.

DEBUG= *integer*
Specifies which level of tracing should be activated. For a description of possible DEBUG values, refer to &diag..

If you decide to change the DEBUG level, you do not need to first set the trace off and then set it on again with the new level. You can simply provide a new level on another TRACEON statement. The SCU does not consider the processing of two successive TRACEON statements as an error.

Example of the TRACEON Statement

Figure 103 shows an example of the TRACEON control statement.

```
TRACEON DEBUG=2,DBD=SKILLDB,SEG=ROOT
```

Figure 103. Example of TRACEON

Output Messages and Return Codes of the SCU

Error messages for the SCU have a prefix of EKYS. For more information on SCU messages and codes, see IMS DPROP Messages and Codes.

The SCU. provides the following return codes:

Code Meaning

0 All requested changes have executed successfully.

4 Warning messages were issued for one or more of the following reasons:

- Some propagation requests were already in the requested mode. For example, an ACTIVATE control statement was provided for a propagation request whose state was already marked active. In this case, the return code of 4 can be handled as a return code of 0. It warns that changing the state of the propagation request is unnecessary.
- An IMS database could not be set to READON or READOFF mode because:
  - DBRC has not been defined for share control
  - The IMS database is not registered in RECON
  - The IMS database is a DEDB
- The SCU does not know whether a status change was performed in an orderly fashion. For example, when propagation requests are performing IMS-to-DB2 propagation and:
  - DBRC has not been defined for share control
  - A propagated IMS database is not registered in RECON.

8 Some or all of the requested changes could not be made.

This return code indicates that no valid control statement is found in //SCUIN data set, or the operation replies TERM on outstanding WTOR console message.

16 A severe error was detected.

This return code indicates that no valid control statement is found in //SCUPLAN data set or when a termination error message is issued.

Part 5. LOG-ASYNC Utilities

Chapter 8. Status Change Utility (SCU) for LOG-ASYNC

This topic describes the use of the Status Change Utility (SCU) for asynchronous, log-based propagation (LOG-ASYNC). The SCU can also be used for asynchronous, MQSeries-based (MQ-ASYNC) propagation and for synchronous propagation; use of the SCU for MQ-ASYNC and for synchronous propagation are described in Chapter 17, “Status Change Utility (SCU) for MQ-ASYNC ,” on page 401 and Chapter 7, “Status Change Utility (SCU) for Synchronous Propagation ,” on page 175, respectively.

For LOG-ASYNC, the SCU supports the timestamp marker facility (TSM facility) control statements, the Receiver control table (RCT) control statements, and propagation request control table (PRCT) control statements. You cannot activate or deactivate propagation requests in an asynchronous system; all propagation requests are considered active. You can use the SCU to:

- Create, delete, and assign timestamp markers.
- Maintain the Receiver control table and the propagation request control table.
- Check the status of Receivers and propagation requests in the Receiver control table and the propagation request control table.
- Set and reset the read-only status of full-function IMS databases.
- Change the error option of a propagation request.
- Initialize the master table of the IMS DPROP directory file.
- Build or initialize the VLF objects containing IMS DPROP control information.
- Display the amount of virtual storage required by the VLF objects of this virtual system.

We strongly encourage you to implement DBRC share control, and register the propagated IMS databases used for production. This will enable the SCU support that can set Full Function IMS DBs in read-only mode.

The following topics provide additional information:

- “Input and Output”
- “JCL Requirements ” on page 214
- “Use of the SCU for LOG-ASYNC Propagation ” on page 222
- “Control statements for LOG-ASYNC propagation ” on page 223
- “Output Messages and Return Codes” on page 242

Input and Output

Figure 104 on page 214 illustrates the input to, and the output created by the SCU.

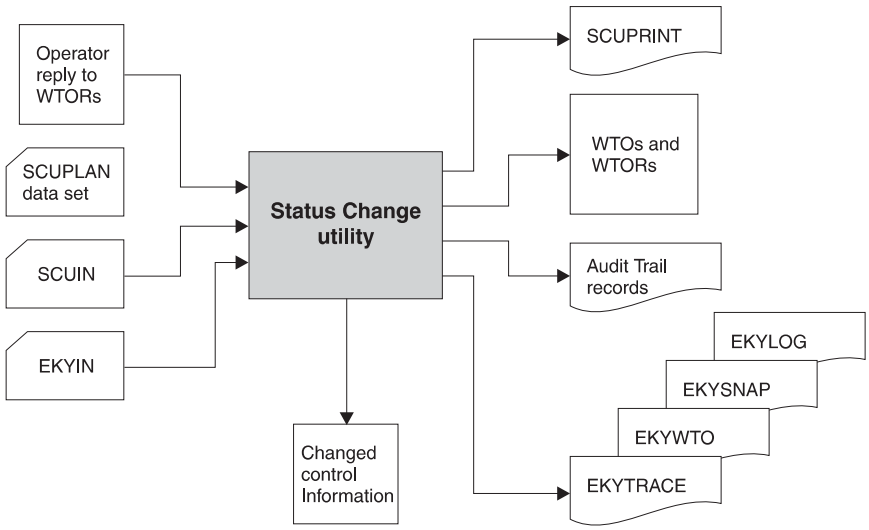

Figure 104. Status Change utility: Input and Output

The input to the SCU is:

- The //SCUIN data set, which contains the control statements
- The //SCUPLAN data set
- Operator replys to WTORs

  If the SCU is in a wait state because of conflicting database authorizations, the operator can use WTOR to:
  - Enter LIST to list the conflicting authorizations.
  - Enter TERM to terminate execution of the SCU.
- The //EKYIN data set

The output from the SCU is:

- Changed control information:
  - The timestamp records in the SCF
  - The propagation request control table and Receiver control table in the IMS DPROP directory
  - other control information in the DPROP Directory tables
  - The virtual lookaside facility (VLF) copy of IMS DPROP information
  - The access mode of IMS databases in RECON data sets
- The //EKYTRACE, //EKYWTO, //EKYSNAP and //EKYLOG files
- Audit trail records, see *IMS DPROP Messages and Codes* for listings

JCL Requirements

This topic contains samples of the JCL that executes the SCU. There are JCL samples to start the SCU, and execute RCT and PRCT commands, TSM commands, and quiesce commands. Table 31 on page 215 lists the JCL samples for each function, and the page number in this topic where you will find them.

Table 31. JCL to Run the SCU

To Do This	Use This JCL	On Page
Start the SCU	EKYUSCUP	215
Call EKYUSCUP to run Receiver control table and propagation request control table commands	EKYUSCUJ	217
Call EKYUSCUP to run timestamp marker facility commands	EKYUSCTJ	219
Call EKYUSCUP to run timestamp marker facility CREATETSM QUIESCE commands	EKYUSCQJ	220

EKYUSCUP—JCL to Execute the SCU

Figure 105 on page 216 is a sample EKYUSCUP, the JCL procedure EKYUSCUP used to execute the SCU.

```
//*-------------------------------------------------------------------*
//*   PROCEDURE EKYUSCUP TO EXECUTE THE SCU                          *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************

//*-------------------------------------------------------------------*
//*                                                                  *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S NEED:         *
//*  -----------------------------------------------------           *
//*                                                                  *
//*  1111111   - DPROP RESLIB HLQ(S)                                 *
//*                                                                  *
//*-------------------------------------------------------------------*
//EKYUSCU      PROC EKYPREF=1111111
//*-------------------------------------------------------------------*
//SCU          EXEC     PGM=EKYSCU00
//*******************************************************************
//* NOTE: IN ASYNCH MODE, A STATUS FILE IS NOT REQUIRED AND THE      *
//* FOLLOWING DD STATEMENT SHOULD BE REMOVED.                        *
//*                                                                  *
//EKYSTATF  DD   DSN=&EKYPREF..EKYSTATF,DISP=SHR
//*******************************************************************
//*
//EKYRESLB     DD       DISP=SHR,
//             DSN=&EKYPREF..EKYRESLB
//SCUPRINT     DD       SYSOUT=*
//SYSPRINT     DD       DSN=&&SYSPRINT,UNIT=SYSDA,SPACE=(CYL,1)
//SYSIN        DD       DSN=&&SYSIN,UNIT=SYSDA,SPACE=(CYL,1)
//*-------------------------------------------------------------------*
//*******************************************************************
//*-----------------------------------------------------------------
//*       ADD THE RECON DD STATEMENTS HERE IF YOU ARE NOT USING
//*       DYNAMIC ALLOCATION
//*----------------------------------------------------------------
//*       &RECON1   = --+
//*       &RECON2   =   '  -  DBRC RECON DATASETS
//*       &RECON3   = --+
//*
//*******************************************************************
//*-----------------------------------------------------------------
```

Figure 105. Sample JCL Procedure EKYUSCUP

Before you submit the JCL, replace 1111111 with IMS DPROP RESLIB high-level qualifiers.

The DD statement //EKYRESLB is common to many IMS DPROP components and is described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The remaining DD statements used in the EKYUSCUP JCL are:

//EKYSTATF DD Statement
When using the SCU for LOG-ASYNC, remove this DD Statement from the EKYUSCUP procedure.

//SCUPRINT
Contains the print output from the SCU. The print output includes a list of all SCU control statements, confirmation that they have been executed, and a list of SCU error messages.

//SYSPRINT
Contains the print output of the IMS DBRC utility called by the SCU.

This DD statement must be allocated to a temporary data set and *not* allocated as a SYSOUT file.

//SYSIN
Contains input commands generated by the SCU to the IMS DBRC utility. The SCU calls the IMS DBRC utility.

This DD statement must be allocated to a temporary data set and *not* allocated as a SYSIN file.

//EKYSCF
Describes the Selector, which contains the IMS source for the propagation groups. Figure 109 on page 219 shows an example of the JCL for the EKYSCF DD statement.

//EKYTSMF
Describes the temporary data set used during processing of the CREATETSM QUIESCE control statement. This data set contains the database names and the TSMID specified on the control statement, together with the quiesce timestamp for the databases. EKYT208X later reads this information from the data set and writes it to the SCF.

Figure 110 on page 221 shows an example of the JCL for the EKYTSMF DD statement.

The SCU uses DBRC and therefore requires access to the DBRC RECON data sets. If you use DBRC without dynamic allocation, you must add the following DD statements to the JCL.

```
//RECON1 DD DSN=recon1,DISP=SHR
//RECON2 DD DSN=recon2,DISP=SHR
//RECON3 DD DSN=recon3,DISP=SHR
```

EKYUSCUJ—JCL to Execute EKYUSCUP

Figure 106 on page 218 is an example of EKYUSCUP, the JCL used to call the SCU for Receiver control table and propagation request control table commands.

```
//JJJJJJJ  JOB  *** VALID JOB CARD DETAILS ***
//*-------------------------------------------------------------------*
//*    JOB TO EXECUTE THE SCU PROCEDURE (EKYUSCUP)                    *
//*                                                                   *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S NEED:          *
//*  -------------------------------------------------------          *
//*                                                                   *
//*  JJJJJJJ   - JOB NAME                                             *
//*  1111111   - SCU CONTROL STATEMENT(S) AS REQUIRED                 *
//*  2222222   - DB2 SUBSYSTEM NAME                                   *
//*  3333333   - NAME OF THE SCU PLAN                                 *
//*                                                                   *
//*-------------------------------------------------------------------*
//EKYUSCU      EXEC EKYUSCUP
//SCUIN     DD   *
1111111
/*
//SCUPLAN   DD   *
SYSTEM=2222222,PLAN=3333333;
//*-------------------------------------------------------------------*
```

Figure 106. EKYUSCUJ: Sample JCL to Run the SCU for the Receiver

//SCUPLAN DD Statement

Identifies the file that contains a control statement that identifies the name of the DB2 plan that executes the SCU. The //SCUPLAN file is input to the SCU.

In a multiple site environment, where the SCU runs only on the IMS side, you do not need to specify the name of the DB2 plan because the SCU will not need to access DB2.

Figure 107 shows the syntax of the //SCUPLAN file.

```
►►──PLAN=plan_name──┬──────────────────────┬──;──────────────────────────►◄
                    └─,SYSTEM=system_name──┘
```

Figure 107. The //SCUPLAN file

PLAN= *plan_name*
Names the DB2 plan that executes the SCU.

SYSTEM= *system_name*
Identifies the DB2 subsystem. If you do not use this keyword, the SCU uses the default DB2 subsystem identifier in module DSNHDECP.

Figure 108 shows two examples of the //SCUPLAN control statement. The first example shows that DB2 plan TEST01 executes the SCU on the subsystem identified in DSNHDECP. The second example shows that DB2 plan TEST01 executes the SCU on system TEST02.

```
PLAN=TEST01
PLAN=TEST02,SYSTEM=TEST02;
```

Figure 108. //SCUPLAN control statement Examples

//SCUIN
Contains the stream of SCU control statements.

EKYUSCTJ—JCL To Execute the SCU for Timestamp Marker Commands

Figure 109 is an example of the JCL for EKYUSCTJ, which calls EKYUSCUP to process timestamp marker facility commands.

```
//JJJJJJJ JOB  *** Valid job card details ***
//*-------------------------------------------------------------------*
//*  JOB TO EXECUTE THE SCU PROCEDURE (EKYUSCUP)                      *
//*       TO PROCESS TSMF CONTROL STATEMENTS:                         *
//*              CREATETSM STOP                                       *
//*              ASSIGNTSM                                            *
//*              DELETETSM                                            *
//*                                                                   *
//* NOTE: USE JOB EKYUSCQJ TO PROCESS TSMF CONTROL STATEMENT:         *
//*              CREATETSM QUIESCE                                    *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S NEED:          *
//*  -------------------------------------------------------          *
//*                                                                   *
//*  JJJJJJJ   - JOB NAME
//*  1111111   - TSMF CONTROL STATEMENT(S) AS REQUIRED                *
//*  2222222   - SELECTOR CONTROL FILE HLQ                            *
//*                                                                   *
//*-------------------------------------------------------------------*
//EKYUSCU      EXEC EKYUSCUP
//SCUIN     DD   *
1111111
/*
//EKYSCF       DD       DSN=2222222.EKYSCF,DISP=SHR
//*******************************************************************
//*-------------------------------------------------------------------*
```

Figure 109. EKYUSCTJ: Sample JCL to Run the SCU for Timestamp Marker Commands

EKYUSCQJ—JCL to Execute the SCU for CREATETSM QUIESCE

Figure 110 on page 221 is an example of the JCL for EKYUSCQJ, which calls EKYUSCUP to process the timestamp marker facility CREATETSM QUIESCE command.

```
//JJJJJJJ  JOB  *** VALID JOB CARD DETAILS ***
//*-------------------------------------------------------------------*
//*  JOB TO EXECUTE THE SCU PROCEDURE (EKYUSCUP)                      *
//*      TO PROCESS TSMF CONTROL STATEMENT:                           *
//*              CREATETSM QUIESCE                                    *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S NEED:          *
//*  -------------------------------------------------------          *
//*                                                                   *
//*  JJJJJJJ   - JOB NAME                                             *
//*  1111111   - TSMF CONTROL STATEMENT(S) AS REQUIRED                *
//*  2222222   - SELECTOR CONTROL FILE HLQ                            *
//*  3333333   - DPROP RESLIB HLQ                                     *
//*                                                                   *
//*-------------------------------------------------------------------*
//EKYUSCU      EXEC EKYUSCUP
//SCUIN     DD   *
1111111
/*
//EKYSCF       DD       DSN=2222222.EKYSCF,DISP=SHR
//*-------------------------------------------------------------------*
//*******************************************************************
//* NOTE: THE FOLLOWING DD STATEMENT IS ONLY REQUIRED FOR THE        *
//*       'CREATETSM QUIESCE' FUNCTION.  IT IS USED TO STORE THE     *
//*       QUIESCE TIMESTAMP AND THE 'CREATETSM QUIESCE' CONTROL      *
//*       STATEMENT PARAMETERS.  IT IS THEN USED AS INPUT TO         *
//*       EKYT208X TO ACTUALLY PERFORM THE UPDATE TO THE SCF.        *
//*       THIS ALLOWS THE SCU 'READOFF' CONTROL STATEMENT TO BE      *
//*       PROCESSED AS QUICKLY AS POSSIBLE THEREBY RELEASING THE     *
//*       RELEVANT IMS DATABASES FROM THEIR READ ONLY STATUS         *
//*       (WHICH WOULD HAVE TYPICALLY BEEN SET USING THE SCU         *
//*       'READON' CONTROL STATEMENT).                               *
//*                                                                  *
//EKYTSMF      DD       DSN=&&EKYTSMF,DISP=(,PASS),UNIT=SYSDA,
//             SPACE=(TRK,(1))
//*******************************************************************
//*-------------------------------------------------------------------*

//*******************************************************************
//* NOTE: THE FOLLOWING STEP IS REQUIRED WHEN EXECUTION OF THE       *
//*       PREVIOUS STEP INCLUDES A EKYTSMF DD STATEMENT. THIS STEP   *
//*       PROCESSES THE STATEMENTS IN THE EKYTSMF FILE TO CREATE THE *
//*       DESIGNATED TIMESTAMP RECORD IN THE SCF.                    *
//*                                                                  *
//*                                                                  *
//CREATTSM     EXEC  PGM=EKYT208X,COND=(04,LT)
//EKYRESLB     DD       DSN=3333333.EKYRESLB,DISP=SHR
//EKYTSMF      DD       DSN=&&EKYTSMF,DISP=(OLD,DELETE)
//EKYSCF       DD       DSN=2222222.EKYSCF,DISP=SHR
//SCUPRINT     DD       SYSOUT=*
//SYSPRINT     DD       SYSOUT=*
//*******************************************************************
//*-------------------------------------------------------------------*
```

Figure 110. EKYUSCQJ: Sample JCL to Run the SCU for CREATETSM QUIESCE

Use of the SCU for LOG-ASYNC Propagation

For an Asynchronous Log-based (LOG-ASYNC) propagation, the SCU supports the timestamp marker facility (TSM facility) control statements, the Receiver control table (RCT) control statements, and propagation request control table (PRCT) control statements. You cannot activate or deactivate propagation requests in an asynchronous system; all propagation requests are considered active. You can use the SCU to:

- Create, delete, and assign timestamp markers.
- Maintain the Receiver control table and the propagation request control table.
- Check the status of Receivers and propagation requests in the Receiver control table and the propagation request control table.
- Set and reset the read-only status of full-function IMS databases.
- Change the error option of a propagation request.
- Initialize the master table of the IMS DPROP directory file.
- Build or initialize the VLF objects containing IMS DPROP control information.
- Display the amount of virtual storage required by the VLF objects of this virtual system.

Timestamp Marker Facility (TSMF)

The timestamp marker facility (TSMF) provides start and stop times to run the Selector. By specifying start and stop times, you delimit the source input for each group to be propagated. Timestamp markers are essential to the integrity of the data supplied to the target. The TSMF supports the following control statements:

- ASSIGNTSM
- CREATETSM
- DELETETSM

The Receiver Control and Propagation Request Control Tables

Use the following SCU control statements to populate and update the tables.

- ASSIGNPR
- CREATEREC
- DELETEPR
- DELETEREC

Together, the Receiver control table and the propagation request control table contain information about propagation requests assigned to particular Receivers, the status of the Receivers, and the status of the PRDSs for the Receivers. The Receiver control table and the propagation request control tables are IMS DPROP directory tables. See Chapter 6, “IMS DPROP Directory Tables,” on page 139 for information on the IMS DPROP directory tables.

Setting and Resetting the Read-Only Status of IMS Databases

IMS DPROP and the SCU allow to set IMS Databases in Read-only mode and to wait until all IMS subsystems release their database update authorization.

This allows to quiesce the IMS databases, when you want to reflect in the target DB2 tables a clean Source System Point-In-Time. And this can be used to prevent IMS Database updates during an IMS Database Extract.

This SCU support is provided only for registered, full-function databases if share control is in effect. If share control is in effect, you can use the SCU to set one of the following two database states:

- Read-only (READON)
- Read-only-off (READOFF)

IMS records these states in the DBRC RECON data sets.

When you call the SCU to set read-only, it calls the IMS DBRC utility with the CHANGE.DB READON command. The SCU returns with a zero return code when all IMS subsystems release their database authorization. You can then assume that no IMS subsystems are updating the database and it is safe to extract the database.

To set read-only-off for a database, the SCU calls the IMS DBRC utility with the CHANGE.DB READOFF command. When the database state is returned to read-only-off, the IMS subsystems can again update the databases.

Control statements for LOG-ASYNC propagation

Table 32 describes the SCU control statements that are used for an Asynchronous Log-Based (LOG-ASYNC) propagation.

Table 32. List of SCU control statements

Control statement	Object affected by the control statement	Site the SCU must run on	Where the state of the object is recorded
ASSIGNPR	PR	Receiver	Propagation request control table in IMS DPROP directory
ASSIGNTSM	Timestamp	Selector	0302 group/database start timestamp record in SCF
CREATETSM	Timestamp	Selector	0202 database/quiesce timestamp record in SCF 0305 group stop timestamp record in SCF
CREATEREC	Receiver	Receiver	Receiver control table in IMS DPROP directory
DELETEPR	PR	Receiver	Propagation request control table in IMS DPROP directory
DELETEREC	Receiver	Receiver	Receiver control table in IMS DPROP directory
DELETETSM	Timestamp	Selector	0202 database/quiesce timestamp record in SCF 0305 group stop timestamp record in SCF
DISPLAY	VLF	Receiver	N/A
ERROPT	PR	Receiver	PR Mapping table in IMS DPROP directory
INIT	Master table/VLF	Receiver	Master table in IMS DPROP directory/VLF objects

Table 32. List of SCU control statements (continued)

Control statement	Object affected by the control statement	Site the SCU must run on	Where the state of the object is recorded
READOFF	IMS database	Selector	RECON data sets
READON	IMS database	Selector	RECON data sets

Multi-site Environments

In a multi-site environment, some control statements are specific to the Selector or Receiver site.

Selector Site control statements

Selector site control statements have no dependency on DB2. Control statements that require that the SCU runs on the Selector site are:

- Control statements that implement TSMF:
  - ASSIGNTSM
  - CREATETSM
  - DELETETSM
- Control statements that control whether the IMS subsystems have update access to the databases:
  - READON
  - READOFF

Receiver Site control statements

The IMS DPROP directory tables exist on the Receiver site. Control statements that require the SCU runs on the Receiver site are those that maintain the Receiver control table and the propagation request control table:

- ASSIGNPR
- CREATEREC
- DELETEPR
- DELETEREC

These control statements set the DB2 tables to DB2 read-write mode.

Syntax of the SCU control statement

Figure 111 on page 225 describes the syntax of the SCU control statement.

dwidth=150'.

```
►►──Statement name──────────────────────────────────────────────────────────►◄

                  ┌──,──┐
►►──┬─PR=(──┬─pr_id──┴─)──────────────────────────────────────────────┬──;──►◄
    │       ▼                                                         │
    │       ┌──,──┐                                                   │
    ├─PRSET=(──┬─prset_id──┴─)──────────────────────────────────────────┤
    │       ▼                                                         │
    │       ┌──,──┐                                                   │
    ├─DBD=(──┬─dbd_name──┴─)──────────────────────────────────────────┤
    │                                  ┌──,──┐          ┌──,──┐      │
    ├─DBD=(─dbd_name─),SEG=(──┬─seg_name──┴─),PRSET=(──┬─prset_id──┴─)─┤
    │                         ▼                        ▼             │
    └─ALL───────────────────────────────────────────────────────────────┘
```

Figure 111. Syntax of the SCU control statement

The Propagation Request Identification Block

Many of the SCU control statements have the same combination of keywords. The combination of the PR, PRSET, DBD, SEG, and ALL keywords comprise a *PR identification block* that you will see repeated in many of the SCU control statements. Use the propagation request identification block to describe to which propagation requests to apply the SCU control statement.

PR=(*prid1,prid2,...* **)**
Names a propagation request or a list of propagation requests.

PRSET=(*prset1,prset2,...* **)**
Names a propagation request set or a list of propagation request sets. By using this keyword alone or together with a DBD keyword, you can limit the scope of the control statement to the propagation requests that belong to the specified PRSET.

DBD=(*database1,database2,...* **)**
Names a physical IMS database, or a list of physical IMS databases.

When you use this keyword, the SCU applies the control statement to all propagation requests from the specified IMS databases for IMS to DB2 propagation.

DBD= *database* **, SEG=(***segname1,segname2,...***), PRSET=(***prset1,prset2,...***)**
Names a physical IMS segment type, or a list of IMS segment types.

If you use the SEG keyword, you must specify one physical IMS database name on the DBD keyword. When you specify the combination of DBD= and SEG= keywords, the SCU applies the control statement to all propagation requests from the specified IMS segment types.

Do not specify the name of internal segments in the SEG= keyword. Internal segments are not IMS segments; they represent structures that are embedded within a containing IMS segment, that are propagated by mapping case 3 propagation requests.

When requesting an SCU function for propagation requests of internal segments, specify either:
- The propagation request IDs on the PR= keyword
- The name of the containing IMS segment in the SEG= keyword

ALL
Specifies all propagation requests of a IMS DPROP system.

Internal Segments

If you specify the name of an IMS segment that contains internal segments in the SEG= keyword, the SCU control statement is applied to:
- All propagation requests for the internal segments
- All propagation requests for the containing IMS segment

Path Data

If an IMS segment specified in the SEG= keyword contains fields used as path data, the SCU control statement is:
- Applied *only* to those propagation requests that propagate the identified segment as an entity or extension segment, and to those propagation requests that propagate internal segments contained in the identified IMS segment.
- *Not* applied to those propagation requests that propagate dependent segments and include path data from the identified segment.

ASSIGNPR

Use the ASSIGNPR control statement to assign a propagation request or set of propagation requests to a Receiver. Before you assign a propagation request to a Receiver, you must use the SCU control statement CREATEREC to create a Receiver row in the Receiver control table. See “CREATEREC ” on page 230. See “The Propagation Request Identification Block ” on page 225 for a description of the propagation request identification block.

Rules Governing the Assignment of Propagation Requests

For an initial request, the propagation request must exist in the propagation request table of the IMS DPROP directory and the Receiver to which the propagation request is assigned. The Receiver must not be running.

To reassign a propagation request to a different Receiver, ensure that the propagation request is assigned to the old Receiver and that neither the old or new Receiver is executing at the time you reassign the request. The values of GROUPID, PRDSSEQ PRDSSTAT, and UOWID, are compared to the corresponding rows in the Receiver control tables for the old and new Receivers. If the values are not the same, and the new Receiver status and position values are not blank, you can reassign a propagation request only if MERGE=FORCE.

Syntax of the ASSIGNPR control statement

Figure 112 on page 227 shows the syntax of the ASSIGNPR control statement.

```
                  ┌──,──┐
                  ▼     │
►►─ASSIGNPR──┬─PR=(──pr_id──)──────────────────────────────────────►
             │
             │          ┌──,──┐
             │          ▼     │
             ├─PRSET=(──prset_id──)──────────────────────────────
             │
             │          ┌──,──┐
             │          ▼     │
             ├─DBD=(──dbd_name──)────────────────────────────────
             │
             │                   ┌──,──┐
             │                   ▼     │
             ├─DBD=(─dbd_name─),SEG=(──seg_name──)──────────────
             │
             └─ALL──────────────────────────────────────────────┘

►─,RECNAME=receiver_name──┬──────────────────────────┬──┬──────────────────┬──;──────────►◄
                          └─,OLDREC=receiver_name────┘  └─,MERGE=FORCE──────┘
```

Figure 112. The ASSIGNPR control statement

RECNAME= *receiver_name*
Names the Receiver to which a propagation request or set of propagation requests is assigned.

OLDREC= *receiver_name*
Names the Receiver to which the propagation requests are currently assigned. Use OLDREC together with RECNAME to reassign a propagation request or set of propagation requests to a new Receiver. The name you specify for RECNAME identifies the new Receiver.

MERGE= *FORCE*
Allows you to override the rules that govern whether a set of propagation requests can be reassigned from one Receiver to another. See “Rules Governing the Assignment of Propagation Requests” on page 226. If you want to bypass the merge rules, specify MERGE=FORCE.

Examples of the ASSIGNPR control statement

Example 1: The example in Figure 113 assigns three propagation requests to Receiver rcvr1 after you use CREATEREC to create a row for the Receiver in the Receiver control table.

```
ASSIGNPR=(PR1,PR2,PR3),RECNAME=rcvr1
```

Figure 113. ASSIGNPR Example 1

Example 2: The example in Figure 114 on page 228 reassigns *PR3* to a new Receiver, *recover*. The value for OLDREC is rcvr1, the name of the Receiver to which PR3 is currently assigned. You might use this example after Receiver *rcvr1* ran a few times and failed due to the unavailable status of the target table for PR3.

```
ASSIGNPR=(PR3),RECNAME=recover,OLDREC=rcvr1
```

Figure 114. ASSIGNPR Example 2

Example 3: The example in Figure 115 reassigns PR3 to the original Receiver, *rcvr1*. You might use this example after Receiver rcvr1 tried to apply updates to a particular PRDS but the target table for PR3 was available. The Receiver, recover, runs until it processes all PRDSs to the particular PRDS.

```
ASSIGNPR=(PR3),RECNAME=rcvr1,OLDREC=recover
```

Figure 115. ASSIGNPR Example 3

ASSIGNTSM

Use the ASSIGNTSM control statement to assign a quiesce timestamp marker as a start timestamp marker for a database within a group.

Syntax of the ASSIGNTSM control statement

Figure 116 shows the syntax of the ASSIGNTSM control statement.

```
►►──ASSIGNTSM──────────────────────────────────────────────────────────►◄

                          ┌─,──────────┐
►►──DBD=dbd_name,GROUP=(──┴─group_id────┴─)──┬─,TIME=──┬─timestamp──────┬──┬──;──────────►◄
                                             │         └─TSM,ID=tsm_id──┘  │
                                             └─,USERTIME=timestamp─────────┘
```

Figure 116. The ASSIGNTSM control statement

ASSIGNTSM

Assigns a quiesce timestamp marker as a start timestamp marker.

DBD= *dbd_name*

Names the database within a group to which the SCU assigns the quiesce timestamp marker as a start timestamp marker.

GROUP=(*group_id1,group_id2,...***)**

Names the groups to which the database belongs.

The database can belong to multiple groups, and the start timestamp marker for the database can be different for different groups. The SCU updates the 0302 group/database start timestamp records that match the database name and group names supplied on this control statement to reflect the quiesce timestamp marker specified.

Each time the SCU assigns a quiesce timestamp marker to a particular 0302 group/database start timestamp record, it sets the newstart flag field in the record to **Y**. The Selector then uses this start timestamp marker for the start of selection for the database in this group. During Selector processing the Selector sets the newstart flag field to **N**. Because the newstart flag is now set to **N**, the Selector does not use this 0302

timestamp marker for start of selection for the data base in this group on subsequent executions of the Selector. The Selector uses the earliest selected 0305 group stop timestamp marker for the start of selection for the database in the group.

TIME= *timestamp* **| TSM,ID=***tsm_id*
Specifies the time value of the quiesce timestamp marker.

The time value can be in either of the following formats:

timestamp
A previously created quiesce timestamp that is assigned as a start timestamp. An example of a timestamp is as follows:

1993-06-01-11.48.23.169268

The format of the timestamp is: *YYYY-MM-DD-HH.MM.SS.NNNNNN*, where:

YYYY	Represents the four digits of the year
MM	Represents the month (include zeros)
DD	Represents the day (include zeros)
HH	Represents the hour (include zeros)
MM	Represents the minutes (include zeros)
SS	Represents the seconds (include zeros)
NNNNNN	Represents the microseconds

TSM,ID= *tsm_id*
A timestamp marker ID associated with a previously created quiesce TSM for this database. It is easier for you to specify a timestamp by referencing its timestamp marker ID.

USERTIME= *timestamp*
Specifies the user-supplied timestamp. You might use this if you know a database was not being updated at a particular time, even though there is no database quiesce timestamp, and you want propagation for the database to start at this time.

Examples of the ASSIGNTSM control statement

Example 1: The example in Figure 117 creates a start timestamp marker by assigning a previously created database quiesce timestamp marker. Specify the database quiesce timestamp marker by either the timestamp value or the timestamp marker ID if one exists. The 26 character timestamp value indicates that updates are selected from the IMSCUST database on June 1, 1993 at 11:48.

```
ASSIGNTSM DBD=IMSCUST,GROUP=GROUP01,TIME=1993-06-01-11.48.23.169268
```

Figure 117. ASSIGNTSM Example 1

Example 2: The example in Figure 118 on page 230 directly creates a start timestamp marker. The timestamp value indicates that updates are selected from the IMSCUST database on June 1, 1993 at 12:30.

```
ASSIGNTSM DBD=IMSCUST,GROUP=GROUP01,USERTIME=1993-06-01-12.30.24.153423
```

Figure 118. ASSIGNTSM Example 2

CREATEREC

Use the CREATEREC control statement to create a new Receiver row in the Receiver control table. You must ensure that a Receiver row exists in the Receiver control table before you assign propagation requests to the Receiver or execute the Receiver.

The CREATEREC control statement also copies information from the current Receiver row to a new Receiver row, for example, during recovery procedures, or when more Receivers are required for a given group.

Syntax of the CREATEREC control statement

Figure 119 shows the syntax of the CREATEREC control statement.

```
►►──CREATEREC──RECNAME=receiver_name,GROUP=group_id──────────────────────────►◄

►►──┬──────────────────────────────────────────────────┬──;──────────────────►◄
    │                   ┌─INIT─┐                       │
    └─,STATUS=──┬───────┴──────┴───────────────────────┴
                └─SPLIT──┬──────────────────────────┬
                         └─,OLDREC=receiver_name────┘
```

Figure 119. The CREATEREC control statement

CREATEREC
Creates a new Receiver row in the Receiver control table.

RECNAME= *receiver_name*
Names the Receiver to use as the primary key for the new Receiver row in the Receiver control table. The Receiver name must not already exist in the Receiver control table.

GROUP= *group_id*
Names the group ID of the group to be processed by the new Receiver.

STATUS= INIT | SPLIT
Indicates whether this is a new Receiver initiation, or if it is a copy of the Receiver for recovery or other purposes.

INIT
Indicates a new Receiver is being initialized. This is the default value for STATUS.

SPLIT
Indicates that Receiver information will be copied from an existing Receiver row to the newly created Receiver row in the Receiver control table. You must specify the old Receiver name on the

OLDREC keyword and ensure that the group ID of the old Receiver matches the group ID of the new Receiver.

OLDREC= *receiver_name*
The old Receiver name. It copies the status information from the old Receiver row to the new Receiver row. This keyword is only required if STATUS=SPLIT. The Receiver must not be active.

Examples of the CREATEREC control statement

Example 1: The example in Figure 120 creates a new Receiver row *rcvr1* in the Receiver control table.

```
CREATEREC RECNAME=rcvr1,GROUP=group1,STATUS=INIT
```

Figure 120. CREATEREC Example 1

Example 2: The example in Figure 121 copies information from existing Receiver row, *rcvr1* in the Receiver control table to a newly created Receiver row, *recover*, in the Receiver control table. STATUS=SPLIT indicates that *rcvr1* is being split for recovery purposes.

```
CREATEREC RECNAME=recover,GROUP=group1,STATUS=SPLIT,OLDREC=rcvr1
```

Figure 121. CREATEREC Example 2

CREATETSM

Use the CREATETSM control statement to create a timestamp marker for propagation selection control. You can create the following timestamps:

DATABASE QUIESCE timestamp
Specifies the time at which an IMS database was in a quiesced, or read-only state.

GROUP STOP timestamp
Specifies the time at which to stop propagation for a group.

Syntax of the CREATETSM control statement

Figure 122 on page 232 shows the syntax of the CREATETSM control statement.

```
►►──CREATETSM──────────────────────────────────────────────────────────►◄

                   ┌─,──────────┐
►►──┬─STOP,GROUP=(──┴─group_id──┴─)──┬──────────────┬──┬──────────────────────┬──;──►◄
    │                                │ ,ID=tsm_id   │  │ ,USERTIME=timestamp  │
    │                   ┌─,──────────┐                │                      │
    └─QUIESCE,DBD=(──┴─dbd_name──┴─)──┴──────────────┴──────────────────────┘
                                      │ ,ID=tsm_id   │
                                      └──────────────┘
```

Figure 122. The CREATETSM control statement

CREATETSM

Creates a quiesce timestamp marker for a database, or a stop timestamp marker for a group.

STOP

Creates a stop timestamp marker for a group.

GROUP=(*group_id1,group_id2,...***)**

Names the groups for which to create a stop timestamp marker. The SCU creates a 0305 group/stop timestamp record in the SCF for each group you specify. If a USERTIME is not specified on this control statement, the current MVS TOD clock value is stored as the timestamp for the 0305 group/stop timestamp record.

ID= *tsm_id*

Identifies a stop timestamp marker to reference the actual timestamp value.

The timestamp marker identifier is an 8 byte alphanumeric value. The first character must be alphabetic.

USERTIME= *timestamp*

Specifies a user supplied timestamp that is used instead of the current MVS TOD clock value. An example is:

1993-06-01-11.48.23.169268

The format of the timestamp is: *YYYY-MM-DD-HH.MM.SS.NNNNNN,* where:

YYYY	Represents the four digits of the year
MM	Represents the month (include zeros)
DD	Represents the day (include zeros)
HH	Represents the hour (include zeros)
MM	Represents the minutes (include zeros)
SS	Represents the seconds (include zeros)
NNNNNN	Represents the microseconds

QUIESCE

Specifies that the database will use a quiesce timestamp marker.

You can use the ASSIGNTSM control statement to assign an existing quiesce timestamp as a start timestamp for a group or database.

The quiesce timestamps are written to the Selector control file (SCF). To avoid keeping IMS databases in read-only state while the Selector is updating the SCF, the SCU processes the QUIESCE control statement in two JCL steps:

1. Save the control statement parameters in a temporary file:
   a. Read CREATETSM QUIESCE control statement parameters.
   b. Check that all databases specified are read-only.
   c. Save the time at which the databases were read-only and write the control statement to a file.
   d. Write an information message informing the user of the quiesce time and the database names.
2. Read the parameters from the file and write to the SCF:
   a. Read the saved CREATETSM QUIESCE parameters and database quiesce timestamps from the file.
   b. Write the quiesce timestamps to the SCF.

See Figure 105 on page 216 for the corresponding JCL.

DBD=(*dbd_name1,dbd_name2,...* **)**
Names the database or databases for which to create a quiesce timestamp marker. The SCU creates a 0202 database/quiesce timestamp record in the SCF for each database you specify, and stores the current MVS TOD clock value as the timestamp for the database/quiesce time records.

You must set each database to READONLY status before creating a quiesce timestamp marker. For a DEDB, the database must be offline because the SCU checks to ensure that each area within the DEDB is unavailable for update.

ID= *tsm_id*
Identifies the actual timestamp value.

The timestamp marker identifier must be unique to the database for quiesce timestamps. It is an 8-byte alphanumeric value. The first character must be alphabetic.

Examples of the CREATETSM control statement

Example 1: The example in Figure 123 creates a stop timestamp marker containing the current MVS TOD clock value for GROUP1.

```
CREATETSM STOP,GROUP=GROUP1,ID=GRP1STOP
```

Figure 123. CREATETSM Example 1

Example 2: The example in Figure 124 on page 234 creates a database quiesce timestamp marker containing the current MVS TOD clock value for the IMSORD database.

```
CREATETSM QUIESCE,DBD=IMSORD,ID=TMIMSORD
```

Figure 124. CREATETSM Example 2

DELETEPR

Use the DELETEPR control statement to delete one or more PR rows from the propagation request control table.

Syntax of the DELETEPR control statement

Figure 125 shows the syntax of the DELETEPR control statement. See “The Propagation Request Identification Block ” on page 225 for a description of the propagation request identification block.

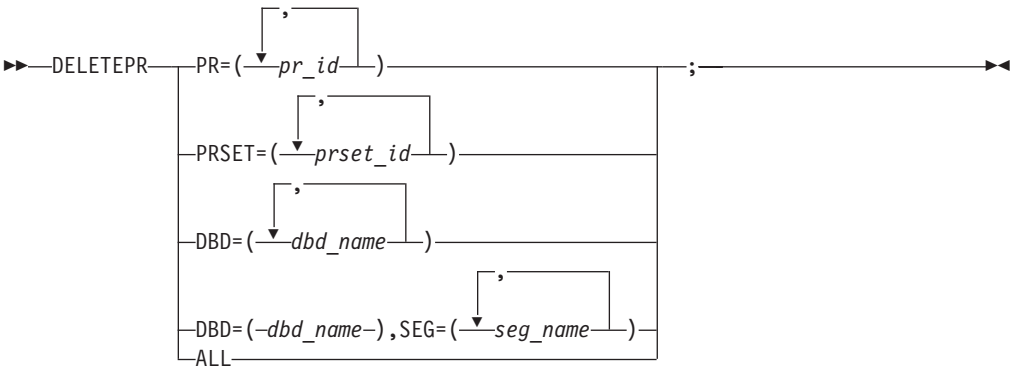

Figure 125. The DELETEPR control statement

Example of the DELETEPR control statement: The example in Figure 126 deletes two rows from the propagation request control table.

```
DELETEPR=(PR103,PR105)
```

Figure 126. DELETEPR Example

DELETEREC

Use the DELETEREC control statement to delete a Receiver row from the Receiver control table. You can delete a Receiver row from the Receiver control table only if there are no propagation requests assigned to the Receiver of that Receiver row.

Syntax of the DELETEREC control statement

Figure 127 on page 235 shows the syntax of the DELETEREC control statement.

```
►►──DELETEREC──RECNAME=receiver_name──;──────────────────────────────────►◄
```

Figure 127. Syntax of the DELETEREC control statement

DELETEREC
Deletes a Receiver row from the Receiver control table.

RECNAME= *receiver_name*
The primary key name the Receiver uses for the Receiver row that is being deleted from the Receiver control table.

Example of the DELETEREC control statement

The example in Figure 128 deletes a Receiver row from the Receiver control table. No propagation requests are assigned to Receiver *recover* so the Receiver row can be deleted.

```
DELETEREC RECNAME=recover
```

Figure 128. DELETEREC Example

DELETETSM

Use DELETETSM to delete timestamps that are no longer required.

Syntax of the DELETETSM control statement

Figure 129 shows the syntax of the DELETETSM control statement.

```
►►──DELETETSM──────────────────────────────────────────────────────────►◄

►►──┬─STOP,GROUP=group_id,──┬─PRIORTO=timestamp──────────┬──;──────────►◄
    │                       ├─PRIORDAY=nn────────────────┤
    │                       └─TIME=timestamp─────────────┘
    └─QUIESCE,DBD=dbd_name,──┬─PRIORTO=──┬─timestamp──────┬──┬
                             │           └─TSM,ID=tsm_id──┘  │
                             └─PRIORDAY=nn──────────────────┘
```

Figure 129. The DELETETSM control statement

DELETETSM
Specifies one or more timestamp markers to delete.

STOP
Specifies that the type of timestamp marker to deleted is a stop timestamp marker for a group.

GROUP= *group_id*
Specifies the group for which the stop timestamp marker is being deleted. The SCU deletes each 0305 group/stop timestamp record that matches the PRIORTO criteria or the PRIORDAY criteria from the SCF.

PRIORTO= *timestamp*
Deletes timestamp markers that have timestamps prior to the specified value.

timestamp
The SCU deletes timestamp markers that have values older than the timestamp specified. However, it does not delete timestamp markers for groups that are marked selected, even if the timestamp marker is older that the specified value.

The Selector uses the most recent group stop time as the group start time the next time you run the Selector.

PRIORDAY= *nn*
Deletes timestamp markers older than *nn* days from the current day.

Specify one of the following values in the *nn* field:

00 Deletes all timestamp markers

01-99 Deletes timestamp markers older than the specified number of days

For example, if you specify the value *03*, timestamp markers three days older than the current date and time are deleted.

The SCU does not delete the most recent group stop time that is marked selected. The Selector uses the most recent group stop time as the group start time for the next Selector execution.

TIME= *timestamp*
Deletes the 0305 group stop timestamp that matches the timestamp specified.

QUIESCE
Specifies that the type of timestamp marker to delete is a quiesce timestamp marker for a DBD.

DBD= *dbd_name*
Specifies the database for which the quiesce timestamp marker is being deleted. The SCU deletes each 0202 database quiesce timestamp record which matches the PRIORTO or PRIORDAY criteria from the SCF.

PRIORTO= *timestamp* **| TSM,ID=***tsm_id*
Specifies the time criteria by which to delete timestamps. Timestamps older than the value specified in *timestamp* are deleted.

TSM,ID= *tsm_id*
Names the timestamp marker identifier for which to delete timestamp markers. Timestamp markers identified by this *tsm_ID* that are older than the value specified in the PRIORTO keyword are deleted.

PRIORDAY= *nn*
Specifies a criteria for deleting timestamp markers. Those that are *nn* days older than the current day are deleted.

Specify one of the following values in the *nn* field:

00 Deletes all timestamp markers

01-99 Deletes timestamp markers older than the specified number of days

For example, if you specify the value *03*, timestamp markers three days older than the current date and time are deleted.

Examples of the DELETETSM control statement

Example 1: The example in Figure 130 deletes all stop timestamp markers for GROUP01 that are over two days old.

```
DELETETSM STOP,GROUP=GROUP01,PRIORDAY=02
```

Figure 130. DELETETSM Example 1

Example 2: The example in Figure 131 deletes all quiesce timestamp markers for the IMSCUST database PRIORTO older than the timestamp of this timestamp marker identifier.

```
DELETETSM QUIESCE,DBD=IMSCUST,PRIORTO=TSM,ID=TMIMSCUST
```

Figure 131. DELETETSM Example 2

DISPLAY

Use the DISPLAY control statement to show the total amount of virtual storage required by the virtual lookaside facility (VLF) objects of this IMS DPROP system.

You can use the information on virtual storage to adjust the MAXVIRT value for the VLF class in the COFVLFxx member. For more information on the MAXVIRT value, see *IMS DPROP Installation Guide*.

Syntax of the DISPLAY control statement

Figure 132 shows the syntax of the DISPLAY control statement.

```
►►──DISPLAY──STATUS──;──────────────────────────────────────────────►◄
```

Figure 132. The DISPLAY control statement

DISPLAY
Displays the operating characteristics of the IMS DPROP system.

STATUS
Checks the status of the IMS DPROP system.

Example of the DISPLAY control statement

Figure 133 shows an example of the DISPLAY control statement.

```
DISPLAY STATUS
```

Figure 133. DISPLAY control statement Example

Example of a DISPLAY control statement Response

Figure 134 on page 238 shows the response to a DISPLAY control statement.

DISPLAY (REQUIRED)
Displays the operating characteristics of the IMS DPROP system.

STATUS (REQUIRED)
Checks the status of the IMS DPROP system.

```
EKYS467I DPROP SYSTEM STATUS: PROPAGATION IS ACTIVE
         DPROP SYSTEM NAME: T003815
         DPROP SYSTEM LEVEL: 110
EKYS468I VLF OBJECT SIZE: 44928 BYTES
         VLF CLASS NAME: PM3
```

Figure 134. DISPLAY Response Example

ERROPT

Use the ERROPT statement to change the error option of a propagation request. The options are IGNORE and BACKOUT.

It might take a few seconds for the RUP to be notified that the error option has been changed. The various propagating regions are not notified of this change simultaneously.

Syntax of the ERROPT control statement

Figure 135 on page 239 shows the syntax of the ERROPT statement. For a description of the propagation request identification block, see “The Propagation Request Identification Block ” on page 225.

```
►►──ERROPT──┬─IGNORE──┬──────────────────────────────────────────────►
            └─BACKOUT─┘

►──┬─,PR=(──┬─pr_id──┬─)──────────────────────────────────────────;──►◄
   │        └──────<──┘
   │
   ├─,PRSET=(──┬─prset_id──┬─)──────────────────────────────────────
   │           └──────<────┘
   │
   ├─,DBD=(──┬─dbd_name──┬─)──┬──────────────────────────────────┬──
   │         └──────<───┘    │                                  │
   │                         └─,PRSET=(──┬─prset_id──┬─)──────┘
   │
   ├─,DBD=dbd_name,SEG=(──┬─seg_name──┬─)──┬──────────────────────┬──
   │                      └──────<───┘    │                      │
   │                                      └─,PRSET=(──┬─prset_id──┬─)──┘
   │
   └─,ALL──────────────────────────────────────────────────────────┘
```

Figure 135. Syntax of the ERROPT control statement

IGNORE | BACKOUT

Use IGNORE to disregard errors issued when propagation cannot be successfully completed.

Use BACKOUT to back out the changes made since the last commit point when errors occur.

Examples of the ERROPT Statement

Figure 136 shows an example of ERROPT with the IGNORE option, and with the BACKOUT option.

```
ERROPT IGNORE,DBD=SKILLS
ERROPT BACKOUT,DBD=SKILLS,SEG=ROOT;
```

Figure 136. ERROPT Examples

INIT

Use the INIT control statement to initialize IMS DPROP control information.

Syntax of the INIT control statement

Figure 137 on page 240 shows the syntax of the INIT control statement.

```
►►──INIT──────────────────────────────────────────────────────────────►◄

►►──┬─DPROP=dpropname─┬──;──────────────────────────────────────────────►◄
    └─VLF─────────────┘
```

Figure 137. The INIT control statement

INIT
Initializes the IMS DPROP directory.

DPROP= *dpropname*
Specifies the name of the IMS DPROP directory for the IMS DPROP system you want to initialize. *dpropname* is defined at installation time.

When you issue the INIT DPROP statement, the SCU verifies that the master table in the IMS DPROP directory is empty. If it is not empty, the SCU writes an error message and stops further processing.

If the directory is empty, the SCU inserts the unique row of the master table of the IMS DPROP directory. This row contains the IMS DPROP system name provided during IMS DPROP installation, and a unique token generated by the SCU.

For more information about the master table, see “The Master Table (DPRMASTER) ” on page 143.

VLF
Builds the VLF objects, which contain IMS DPROP control information.

Use this keyword to populate VLF after an IPL or after a sequence of VLF deactivation and reactivation. All objects of the VLF class belonging to the IMS DPROP system are deleted before being recreated. Use an INIT VLF statement to populate VLF rather than let the RUP populate VLF on a demand basis. Doing so avoids ENQueue conflicts on small IMS DPROP directory tables.

You can also use an INIT VLF after a software error created wrong objects in VLF, or after recovering the IMS DPROP directory tables.

Examples of the INIT control statement

The example in Figure 138 initializes IMS DPROP directory TEST01.

```
INIT DPROP=TEST01
```

Figure 138. INIT Example

READOFF

Use the READOFF control statement to turn off DBRC read-only mode for an IMS full-function database. The READOFF control statement supports full-function IMS databases only if DBRC share control is active and the database is registered in DBRC.

The READOFF control statement resets an IMS database after it was set to read-only to allow database quiesce timestamp markers to be created.

READOFF control statement

Figure 139 shows the syntax of the READOFF control statement.

```
                    ┌─,──────────┐
                    ▼            │
►►──READOFF──DBD=(──dbd_name──────)──;──────────────────────────────────►◄
```

Figure 139. The Syntax of the READOFF control statement

READOFF
Resets the read-only mode of IMS databases.

DBD=(*database1,database2,...* **)**
Names the databases for which read-only mode is reset. Specify the names of a physical IMS databases.

The SCU terminates processing if an error occurs while processing a control statement. To prevent your IMS databases from being left in a read-only state, provide a separate JCL step to set READOFF mode if an SCU control statement fails. See Figure 106 on page 218 for an example of the JCL.

Examples of the READOFF control statement

The example in Figure 140 sets database SKILLDB to READOFF mode.

```
READOFF DBD=SKILLDB
```

Figure 140. READOFF Example

READON

Use the READON control statement to set an IMS full-function database to DBRC read-only mode. The READON control statement supports full-function databases only if DBRC share control is active and if the database is registered in DBRC.

The READON control statement allows database quiesce timestamp markers to be created.

When processing a READON control statement, the SCU waits until the updating IMS subsystems have released their database update authority. This is in contrast to the DBRC CHANGE.DB READON commands: DBRC does not wait until IMS systems release their update authority before returning. For more information on DBRC share control, refer to *IMS/ESA Operations Guide*.

Syntax of the READON control statement

Figure 141 on page 242 shows the syntax of the READON control statement.

```
                  ┌─,──────────┐
                  ▼            │
►►──READON──DBD=(──dbd_name──────)──;──────────────────────────────────►◄
```

Figure 141. Syntax of the READON control statement

READON
Sets the read-only mode of a physical IMS databases.

DBD=(*database1,database2,...* **)**
Specifies the IMS database or list of databases to set to read-only mode.

Example of the READON control statement

The example in Figure 142 sets the databases names SKILLDB and INVENTRY to READON mode.

```
READON DBD=(SKILLDB,INVENTRY)
```

Figure 142. READON Example

Output Messages and Return Codes

Error messages for the SCU begin with EKYS. For more information on SCU messages and codes, refer to*IMS DPROP Messages and Codes*.

The Status Change utility provides the following return codes:

Code	Meaning
0	All requested changes have executed successfully.
4	Warning messages were issued for one or multiple of the following reasons: • The SCU processes READON of READOFF control statements, and an IMS database could not be set to the requested state because: – DBRC has not been defined for share control. – The IMS database is not registered in RECON. – The IMS database is a DEDB. • There was a duplicate propagation request assignment. • There was no propagation request assignment. • The timestamp marker facility is notifying the user of a situation which is not an error. – For DELETETSM: no timestamps matching the criteria were found – For CREATETSM: timestamp already exists
8	Some or all of the requested changes were not made. This return code occurs when: • There is no valid control statement in //SCUIN data set. • The operator replies TERM to an outstanding WTOR console message. • The timestamp marker facility processing detects: – The database not found in the SCF. – The timestamp supplied is not in ISO/DB2 format.
16	A severe error was detected.

This return code occurs when no valid control statement is found in //SCUPLAN data set (when //SCUPLAN is required), or when there is a termination error message.

Chapter 9. Selector Control File (SCF)

The purpose of the Selector control file (SCF) is to tell the Selector what records to collect from the source IMS log files. The SCF contains definitions of the propagation groups that a particular Receiver will process during a particular execution of the Selector. The SCF also contains timestamp information for the propagation records.

The JCL required to administer the SCF is:

EKYGUUP Executes the Group Unload utility (GUU)

EKYUSC1P Executes the SCF Compare utility

EKYUSC2P Executes the SCF Apply utility

EKYUSC2J Calls the EKYUSC2P JCL procedure

EKYUGUUJ Calls the EKYUGUUP JCL procedure

EKYUGUUA Allocates the group definitions file

EKYUSC1A Allocates the SCF control statements file

You can update the SCF either manually or automatically. To update the SCF manually, you create the required SCF control statements and use the SCF Apply utility to apply them to the SCF. However, we recommend you use the automatic method to update the SCF because it validates the input and reduces administration time.

You can update the SCF automatically at either the Selector or Receiver site.

- At the Receiver site, use the SELECT GROUP control statement to specify which propagation groups to store in the SCF. The GUU extracts details of the IMS source for the specified propagation groups from the IMS DPROP directory tables and writes them to the Receiver group definitions file.
- At the Selector site, use the SCF Compare utility to compare the contents of the Receiver group definitions file with the SCF. Then you create SCF control statements to create or modify propagation groups in the SCF. You can modify the control statements before submitting them.

  You use the SCF Apply utility to execute the SCF control statements to bring the propagation group definitions in the SCF into line with those in the IMS DPROP directory.

The following topics provide additional information:

- “How to Update the SCF ” on page 246
- “The Group Unload Utility (GUU) and SCF Utilities ” on page 246
- “Group Unload Utility and SCF Utility Environment ” on page 248
- “Group Unload Utility ” on page 248
- “SCF Compare Utility ” on page 257
- “SCF Apply Utility ” on page 262
- “SCF control statements ” on page 268
- “SCF Record Layout” on page 307

How to Update the SCF

The SCF is created when the IMS DPROP environment is generated. You initially add database records and propagation group records to it by:

- Defining the propagation requests
- Creating Receivers (use the SCU)
- Assigning propagation requests to Receivers
- Updating the SCF to reflect the types of propagation requests being sent to a particular Receiver.

You can add or delete propagation groups from the SCF by using the Group Unload utility and SCF Compare and Apply utilities. You can also add IMS SSIDs by using the SCF Apply utility with the ADDSSID control statement, or delete IMS SSIDs by using the DELSSID or DELGROUP control statement. You might update the SCF when:

- A propagation request in a propagation group changes in such a way that the IMS sources for the propagation request change. For example, the path data changes, new fields or segments are added, existing fields or segments are changed.
- A propagation request is assigned to a Receiver, using the SCU ASSIGNPR control statement. This effectively adds a propagation request to a propagation group.
- A propagation request is re-assigned using the SCU ASSIGNPR control statement, and the new Receiver and the old Receiver process different propagation groups. This effectively moves a propagation request from one propagation group to another.

The SCF control statements are described in this topic starting on page 268.

The Group Unload Utility (GUU) and SCF Utilities

Figure 143 on page 247 shows how the GUU and the SCF utilities are used to administer the SCF.

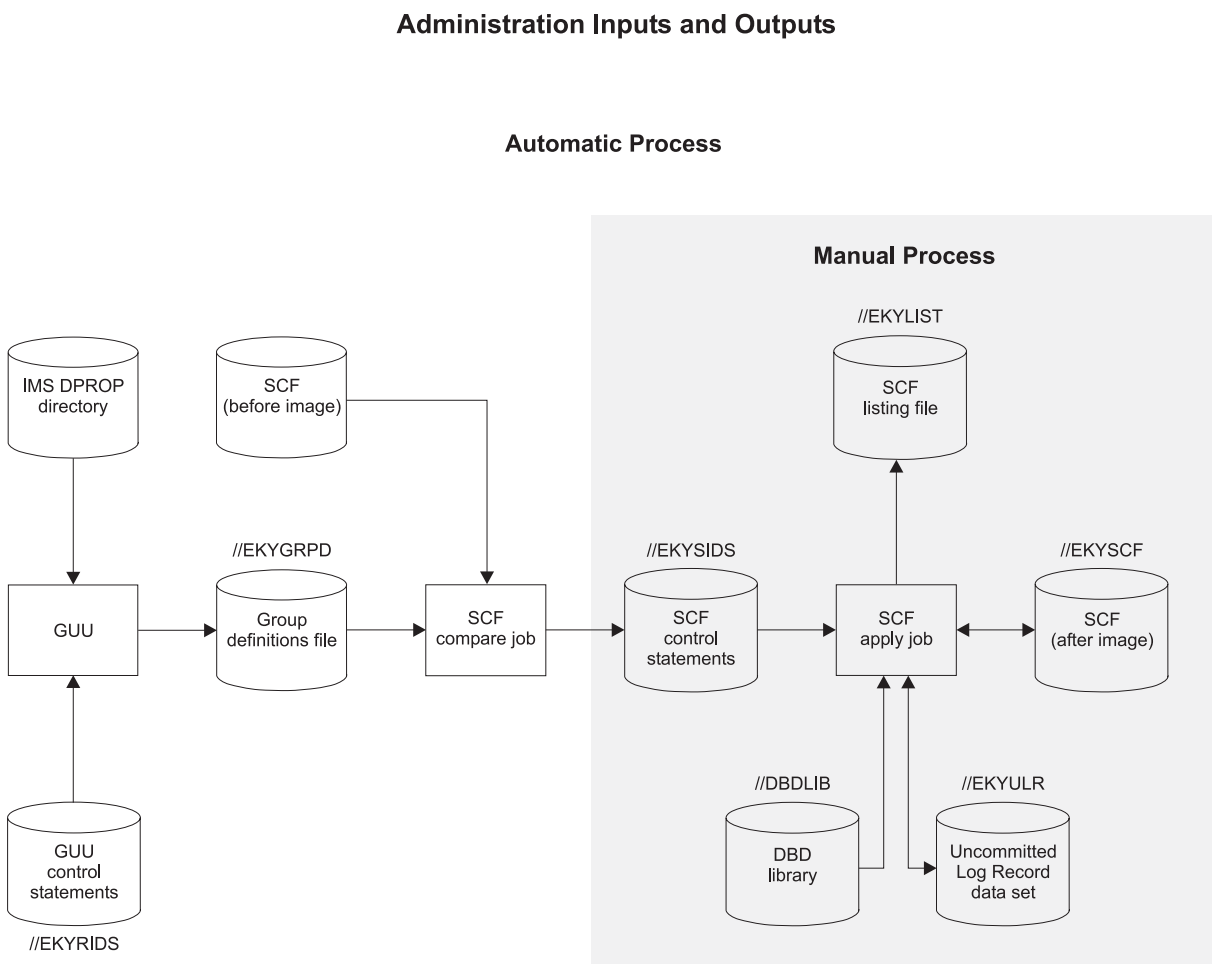

Figure 143. SCF Administration Inputs and Outputs

The following steps demonstrate how the GUU and SCF utilities update the SCF.

1. The GUU extracts details of the sources for one or more Receiver group from the IMS DPROP directory tables and writes them to the Receiver group definitions file.

   The GUU uses the information in the PRCT and RCT IMS DPROP directory tables to determine what propagation requests belong to a specified Receiver group:

   - The RCT contains details of the Receivers that process each propagation group.
   - The PRCT contains details of which Receiver each propagation request is assigned to.

   Because each propagation request is assigned to exactly one Receiver and each Receiver is assigned to exactly one propagation group, each propagation request is a member of exactly one Receiver group. For more information on the RCT and PRCT, see “Control Tables Used by LOG-ASYNC ” on page 161.
2. The SCF Compare utility compares the contents of the Receiver group definitions file with the propagation group records in the SCF and generates SCF control statements to bring the SCF into line with the Receiver group definitions file.

3. The SCF Apply utility uses the SCF control statements to create new propagation groups and to list, modify, or delete existing propagation groups in the SCF.

Group Unload Utility and SCF Utility Environment

The Group Unload utility (GUU) and the SCF Apply and Compare utilities execute as relational applications under MVS/ESA in 31-bit addressing mode (AMODE=31).

The GUU uses the TSO attach to connect to DB2. In the TSO environment you call the GUU with the RUN CP subcommand of the DB2 DSN command.

You can run the utilities whether the Selector and Receiver are installed on the same or different systems. Table 33 explains how to access the group definitions file in each environment.

Table 33. Accessing the Group Definitions file in different GUU Environments

Selector and Receiver	Group Definitions File
Installed on the same MVS image	Automatically accessible by the SCF Compare utility
Installed on different MVS images with shared DASD	Create it on shared DASD
Installed on different MVS images without shared DASD	Transport it from the Receiver site to the Selector site before executing the SCF Compare utility

Group Unload Utility

The GUU extracts details of the IMS sources for one or more groups from the IMS DPROP directory tables and writes them to the group definitions file. The groups processed by the GUU are specified on the GUU SELECT control statement. For the syntax of the SELECT control statement see page 255.

The GUU reads the SELECT control statements from the GUU input control statements data set, //EKYRIDS, to determine what groups must be processed. For each group specified on the SELECT control statements, the GUU:

1. Determines the propagation requests in the group by:
   a. Querying the RCT to determine what Receivers process the group
   b. Querying the PRCT to determine what propagation requests have been assigned to each of the Receivers
2. Determines the IMS sources for each of the propagation requests in the group from the IMS DPROP directory tables. You can obtain this information by querying the propagation request table, SEG table, and FLD Table in the IMS DPROP Directory.
3. Determines the IMS sources for each group as a whole by accumulating the individual IMS sources for the propagation requests in the group and removing any duplicates
4. Writes the IMS sources for each group to the group definitions file, //EKYGRPD.

GUU Input and Output

Figure 144 on page 249 shows the input to and output from the GUU.

GUU Inputs and Outputs

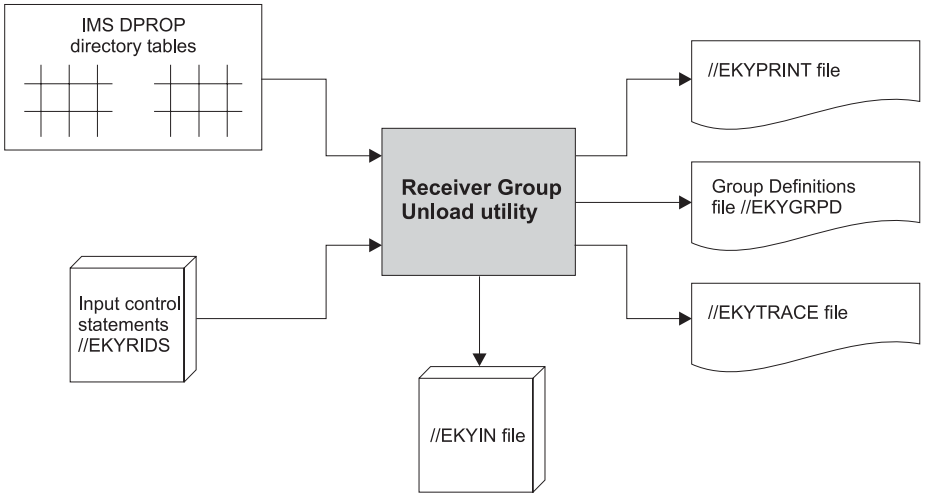

Figure 144. GUU Inputs and Outputs

The input to the GUU consists of:

- The IMS DPROP directory tables
- The GUU input control statements data set, //EKYRIDS, which contains the GUU control statements (SELECT) that specify the groups whose IMS sources are to be written to the group definitions file, //EKYGRPD
- The //EKYIN data set, which can be used to provide the TRACE control statements to trace the activities of the GUU

The output from the GUU consists of:

- The //EKYTRACE file, which receives IMS DPROP trace output
- The //EKYPRINT file, which contains the print output from the GUU
- The group definitions file, //EKYGRPD, which contains the IMS source for the groups selected via the GUU SELECT control statements in the //EKYRIDS data set.

  Figure 145 on page 250 shows the layout of the group definitions file.

```
/*********************************************************/
/*                                                       */
/* Group SGROUP01 contains the following                 */
/* propagation requests:                                 */
/*       TESTPR11                                        */
/*       TESTPR12                                        */
/*       TESTPR31                                        */
/*                                                       */
/*********************************************************/
 GROUP    GROUP=SGROUP01 ;
 DATABASE GROUP=SGROUP01, DBD=TESTDB11 ;
 SEGMENT  GROUP=SGROUP01, DBD=TESTDB11, SEG=TSTSEG11 ;
 DATABASE GROUP=SGROUP01, DBD=TESTDB12 ;
 SEGMENT  GROUP=SGROUP01, DBD=TESTDB12, SEG=TSSEG121 ;
 FIELD    GROUP=SGROUP01, DBD=TESTDB12, SEG=TSSEG121,
 FLDSTRT=00001, FLDLEN=00100, FLDNAME=SEG121F1 ;
 SEGMENT  GROUP=SGROUP01, DBD=TESTDB12, SEG=TSSEG122 ;
 FIELD    GROUP=SGROUP01, DBD=TESTDB12, SEG=TSSEG122,
 FLDSTRT=00001, FLDLEN=00100, FLDNAME=SEG122F1 ;
 SEGMENT  GROUP=SGROUP01, DBD=TESTDB12, SEG=TSSEG123 ;
 FIELD    GROUP=SGROUP01, DBD=TESTDB12, SEG=TSSEG123,
 FLDSTRT=00001, FLDLEN=00100, FLDNAME=SEG123F1 ;
 SEGMENT  GROUP=SGROUP01, DBD=TESTDB12, SEG=TSSEG124 ;
 SEGMENT  GROUP=SGROUP01, DBD=TESTDB12, SEG=TSSEG125 ;
 SEGMENT  GROUP=SGROUP01, DBD=TESTDB12, SEG=TSSEG126 ;
 SEGMENT  GROUP=SGROUP01, DBD=TESTDB12, SEG=TSTSEG12 ;
```

Figure 145. Layout of the Group Definitions File - //EKYGRPD

Specifying Sensitivity Levels for GUU Output

The level of sensitivity you specify for segments and databases in the SCF can affect the performance of both the Selector and the Receiver. The levels of sensitivity are:

Field-level
Define in the group definitions file all fields, segments, and databases that are being propagated.

Segment-level
Remove from the group definitions file all field definitions, and include only the segment and database definitions to be propagated.

Database-level
Include in the group definitions file only the databases to be propagated. Remove field and segment definitions from the group definitions file.

The GUU defines sensitivity at the lowest level possible. Selector performance improves when you raise the level of sensitivity. The lowest level of sensitivity is field level, the highest level is database level.

If you specify field-level sensitivity for a particular segment, the Selector:

- Performs additional checks to identify the log records to process. This increases the Selector's processing time.
- Produces smaller PRDSs than it would for a higher level of sensitivity. This reduces the Receiver's processing time.

For example, if all the fields in a segment are being propagated and are defined in the IMS DPROP directory, the GUU might define field-level sensitivity for the segment in the group definitions file. You can improve the performance of the Selector by changing the sensitivity to segment-level. To specify segment-level sensitivity for a segment, you manually remove all field definitions for the segment from the group definitions file.

Similarly, if all the segments in a database are being propagated and are defined in the IMS DPROP directory, you can improve the Selector performance by specifying database-level sensitivity. To specify database-level sensitivity for a database, you manually remove all segment and field definitions for the database from the group definitions file.

The GUU cannot always determine the role of the segments in user-mapping propagation requests; therefore it provides segment-level sensitivity for all segments including parent and ancestor segments.

If path data is not propagated, segment-level sensitivity is not required for the parent and ancestor segments and you should remove the parent and ancestor segments from the GUU output.

For more details on how to improve Selector performance, see the appropriate Administrators Guide for your propagation mode.

JCL for the GUU

This topic describes the following sample JCL:

- EKYUGUUP, the sample JCL to execute the GUU
- EKYUGUUJ, the sample JCL to call the EKYUGUUP procedure
- EKYUGUUA, the sample JCL to allocate the group definitions file

EKYUGUUP: Sample JCL to Execute the GUU

Figure 146 on page 252 shows the sample JCL procedure EKYUGUUP, which you use to execute the GUU.

Before you invoke the procedure:

- Replace all occurrences of `1111111` with the high-level qualifiers for the IMS DPROP RESLIB.
- Replace all occurrences of `2222222` with the high-level qualifiers for the group definitions file.

```
//*-------------------------------------------------------------------*
//*   PROCEDURE EKYUGUUP TO EXECUTE THE GROUP UNLOAD UTILITY (GUU)    *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S STANDARDS :    *
//*-------------------------------------------------------------------*
//*                                                                   *
//*  1111111   = DPROP RESLIB HIGH LEVEL QUALIFIER (HLQ)              *
//*  2222222   = GROUP DEFINITIONS FILE HLQ                           *
//*                                                                   *
//*-------------------------------------------------------------------*
//*
//EKYUGUUP PROC EKYPREF=1111111,        DPROP RESLIB HLQ
//         EKYGRPD=2222222              GROUP DEFINITIONS FILE
//*
//*-------------------------------------------------------------------*
//*
//RUN     EXEC PGM=IKJEFT01,
//             DYNAMNBR=20
//*
//EKYRESLB DD  DISP=SHR,
//             DSN=&EKYPREF..EKYRESLB
//*-------------------------------------------------------------------*
//*  DATASETS REQUIRED SPECIFICALLY BY THE GROUP UNLOAD UTILITY       *
//*-------------------------------------------------------------------*
//EKYGRPD  DD  DISP=OLD,
//             DSN=&EKYGRPD..EKYGRPD
//*
//EKYPRINT DD  SYSOUT=*
//EKYLOG DD SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//*
//*-------------------------------------------------------------------*
```

Figure 146. EKYUGUUP: Sample JCL to Execute the GUU

The DD statements //EKYRESLB and //EKYPRINT in Figure 146 are common to many IMS DPROP components and are described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The following list describes the other statements in EKYUGUUP.

PROC Statement

Defines the various keyword parameters of the JCL procedure. When you call EKYUGUU, you must provide values for the following keywords. Enclose data set names in single quotes.

EKYPREF The high-level qualifier of the IMS DPROP RESLIB.

EKYGRPD The high-level qualifier of the group definitions file.

EXEC statement

Calls TSO batch to execute the GUU under DB2 with the TSO attach.

//EKYGRPD DD statement
The data set name of the group definitions file the GUU generates.

//SYSTSPRT DD statement
A print file used by the DB2 TSO TMP environment.

EKYUGUUJ: Sample JCL to Call EKYUGUUP JCL

Figure 147 shows the sample JCL for EKYUGUUJ, which calls EKYUGUUP to execute the GUU.

Before you submit the JCL:

- Replace `JJJJJJJ` with valid job card details.
- Replace all occurrences of `1111111` with your DB2 subsystem name.
- Replace all occurrences of `2222222` with your GUU plan name.
- Replace all occurrences of `3333333` with the name of the propagation group to be unloaded.

```
//JJJJJJJ  JOB  ***  VALID JOB CARD DETAILS  ***
//*-------------------------------------------------------------------*
//*       JOB EKYUGUUJ TO EXECUTE THE EKYUGUUP PROCEDURE              *
//*               ( GROUP UNLOAD UTILITY )                            *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*                                                                  *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S STANDARDS :   *
//*-------------------------------------------------------------------*
//*                                                                  *
//*  JJJJJJJ  = JOB NAME                                             *
//*  1111111  = DB2 SUBSYSTEM NAME                                   *
//*  2222222  = NAME OF THE GROUP UNLOAD UTILITY (GUU) PLAN          *
//*  3333333  = NAME OF THE GROUP TO BE UNLOADED                     *
//*                                                                  *
//*-------------------------------------------------------------------*
//*
//*
//RUNGUUP EXEC PROC=EKYUGUUP
//*
//SYSTSIN    DD   *
 DSN SYSTEM(1111111)
 RUN  CP PLAN(2222222)
  EKYQ000X
 END
/*
//EKYRIDS    DD  *
 SELECT GROUP=3333333;
/*
//*
//*-------------------------------------------------------------------*
```

Figure 147. EKYUGUUJ Sample JCL to Call the EKYUGUUP Procedure

The sample in Figure 147 on page 253 contains the following DD statements:

//EKYRIDS DD statement
//EKYRIDS contains the GUU input control statements that specify the groups whose IMS sources are written to the group definitions file. This data set can contain several control statements. For information on the control statements, refer to “GUU SELECT control statement ” on page 255.

//SYSTSIN DD statement
Describes the input data set to TSO batch. //SYSTSIN contains a DSN command, a RUN CP subcommand, the GUU program name that TSO processes as a TSO command, and an END command. This data set is primarily used to execute the GUU as a DB2 application under TSO batch.

EKYUGUUA: Sample JCL to Allocate the Group Definitions File

Figure 148 on page 255 shows the JCL for EKYUGUUA, which allocates the group definitions file. The job contains two steps:

DELETE To delete the group definitions file //EKYGRPD.

ALLOC To allocate a new group definitions file //EKYGRPD.

Before you submit the JCL:

- Replace all occurrences of `JJJJJJJ` with valid job card details.
- Replace all occurrences of `1111111` with the high-level qualifiers for the group definitions file.

```
//JJJJJJJ  JOB  *** VAILD JOB CARD DETAILS  ***
//*-------------------------------------------------------------------*
//*       JOB EKYUGUUA TO ALLOCATE THE GROUP DEFINITIONS FILE         *
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S STANDARDS :    *
//*  ------------------------------------------------------------     *
//*                                                                   *
//*  JJJJJJJ   = JOB NAME                                             *
//*  1111111   = GROUP DEFINITIONS FILE HLQ                           *
//*                                                                   *
//*-------------------------------------------------------------------*
//*
//*****************************************************************
//* DELETE THE GROUP DEFINITIONS FILE
//*****************************************************************
//*
//DELETE      EXEC PGM=IEFBR14
//EKYGRPD     DD   DSN=1111111.EKYGRPD,DISP=(MOD,DELETE),
//            UNIT=SYSDA,SPACE=(800,(10,5))
//*
//*****************************************************************
//* ALLOCATE THE GROUP DEFINITIONS FILE
//*****************************************************************
//*
//ALLOC       EXEC PGM=IEFBR14,COND=(4,LT)
//EKYGRPD     DD   DSN=1111111.EKYGRPD,DISP=(NEW,CATLG),UNIT=SYSDA,
//            DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),SPACE=(800,(10,5))
//*
//*-------------------------------------------------------------------*
```

Figure 148. Sample JCL to Allocate the Group Definitions File

GUU SELECT control statement

Use the SELECT control statement to specify propagation groups to the GUU. The GUU then writes to the group definitions file the IMS source files contained in those propagation groups.

GUU SELECT Syntax

Figure 149 shows the syntax of the SELECT control statement.

```
                    ┌─,──────────┐
                    ▼            │
►►─SELECT─┬─GROUP=(──group_id──┬─)─┬─;──────────────────────────────────►◄
          └─ALL────────────────┘   │
```

Figure 149. The SELECT control statement

ALL
Specifies the IMS source of all the propagation groups in the RCT is to be written to the group definitions file.

GROUP= *(group_id,group_id,.....)*
Specifies the groups whose IMS source is to be written to the group definitions file.

Examples of the GUU SELECT Command

Example 1: The example in Figure 150 generates the IMS sources of all propagation groups defined in the RCT and writes them to the group definitions file, //EKYGRPD.

```
SELECT ALL
```

Figure 150. SELECT control statement: Example 1

Example 2: The example in Figure 151 generates the IMS sources of groups GROUP01 and GROUP02 and writes them to the group definitions file //EKYGRPD.

```
SELECT GROUP=(GROUP01,GROUP02)
```

Figure 151. SELECT control statement: Example 2

GUU Output Messages

GUU warning and error messages begin with EKYQ. For more information on GUU messages and codes, see *IMS DPROP Messages and Codes*.

GUU Return Codes

The GUU return codes are:

0 The sources of all the groups specified on the GUU SELECT control statements have been successfully written to the group definitions file.

4 Warning messages were issued for one or more of the following reasons:
- No propagation requests are assigned to one of the Receivers that process one of the groups selected.
- No propagation requests are assigned to any of the Receivers that process one of the groups selected.
- Field level sensitivity was provided for a segment where a field exit was specified.

8 The sources of all the groups specified on the GUU SELECT control statements could not be successfully written to the group definitions file.

SCF Compare Utility

The SCF Compare utility compares the contents of the group definitions file (//EKYGRPD) to the Selector control file (//EKYSCF) and generates SCF control statements for any differences it finds. Then you use the SCF Apply utility to apply the control statements to the SCF. This ensures the IMS source for the propagation groups in the SCF matches the source for the propagation groups in the IMS DPROP directory.

The SCF Compare utility:

- Reads the group definitions file, //EKYGRPD.
- For each propagation group in the group definitions file:
  1. Compares the definition of the propagation group in the group definitions file with that in the SCF. If they differ, the utility generates the SCF control statements required to match the definition of the propagation group in the SCF with the group definitions file and the IMS DPROP directory.
  2. Writes any SCF control statements generated to the SCF control statements file, //EKYSIDS.
- For each propagation group in the SCF that is not in the group definitions file, it issues a warning indicating that a propagation group exists in the SCF that is not defined in the IMS DPROP directory at the Receiver site. If this group is no longer being propagated, use the DELGROUP control statement and the SCF Apply utility to delete it.

If the GUU runs for only one group because it is the only group in the IMS DPROP directory that has changed, the SCF Compare utility issues warning messages for all other propagation groups in the SCF.

SCF Compare Utility Input and Output

Figure 152 shows the input to and output from the SCF Compare utility.

SCF Compare Utility Input and Output

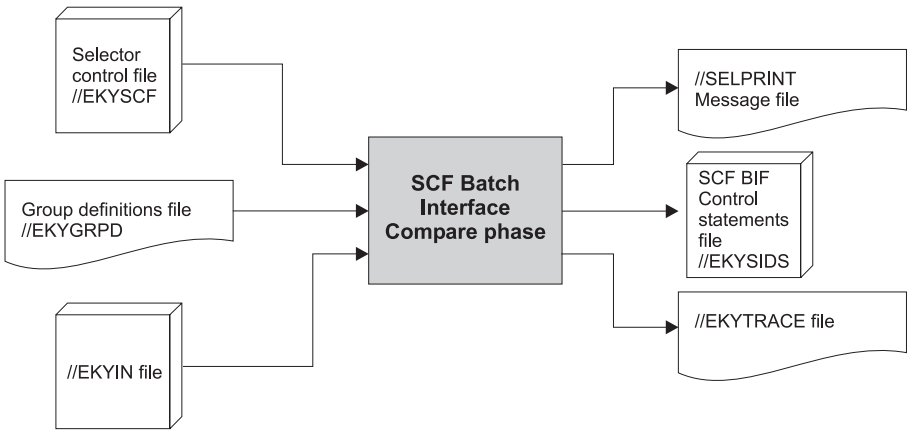

Figure 152. SCF Compare Utility Input and Output

The input to the SCF Compare utility consists of:

- The group definitions file, //EKYGRPD, which contains the IMS source for the propagation groups as defined in the IMS DPROP directory on the Receiver site.
- The SCF, //EKYSCF, which contains the IMS source for the propagation groups as currently defined on the Selector site.
- The //EKYIN data set, which contains the TRACE control statement to trace the activities of the GUU.

The output from the SCF Compare utility consists of:

- The SCF control statements File, //EKYSIDS, which contains the SCF control statements generated by the Compare utility. When these control statements are executed by the SCF Apply utility they change the SCF definition of the IMS source for the propagation groups so they agree with the IMS DPROP directory definitions.
- The //EKYTRACE file, which receives IMS DPROP trace output.
- The //SELPRINT file, which contains the print output from the SCF Compare utility.

SCF Compare Utility JCL

This topic describes the following sample JCL:

- EKYUSC1P, to execute the SCF Compare utility
- EKYUSC1A, to allocate the SCF control statements file

The job you create to call the EKYUSC1P procedure consists of only a simple EXEC statement; therefore, no sample JCL is provided here.

EKYUSC1P: Sample JCL to Execute the SCF Compare Utility

Figure 153 on page 259 shows the sample JCL for EKYUSCF1, which executes the SCF Compare utility.

Before you invoke the procedure:

- Replace all occurrences of `1111111` with the high-level qualifiers for the IMS DPROP RESLIB.
- Replace all occurrences of `2222222` with the high-level qualifiers for the group definitions file.
- Replace all occurrences of `3333333` with the high-level qualifiers for the Selector control file.
- Replace all occurrences of `4444444` with the high-level qualifiers for the SCF control statements file.

```
//*-------------------------------------------------------------------*
//*       PROCEDURE EKYUSC1P TO EXECUTE THE SCF COMPARE UTILITY       *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//* CHANGE THE FOLLOWING TO COMPLY WITH YOUR INSTALLATION STANDARDS:- *
//*-------------------------------------------------------------------*
//*                                                                   *
//*  1111111   - DPROP RESLIB HIGH LEVEL QUALIFIER (HLQ)              *
//*  2222222   - GROUP DEFINITIONS FILE HLQ                           *
//*  3333333   - SELECTOR CONTROL FILE HLQ                            *
//*  4444444   - SCF CONTROL STATEMENTS FILE HLQ                      *
//*                                                                   *
//*-------------------------------------------------------------------*
//*
//EKYUSC1P PROC EKYPREF=1111111,
//         EKYGRPD=2222222,
//         EKYSCF=3333333,
//         EKYSIDS=4444444
//*
//*-------------------------------------------------------------------*
//RUN      EXEC PGM=EKYB700X
//*
//*-------------------------------------------------------------------*
//*  DATASETS REQUIRED SPECIFICALLY BY THE SCF COMPARE UTILITY        *
//*-------------------------------------------------------------------*
//EKYGRPD  DD  DISP=SHR,
//             DSN=&EKYGRPD..EKYGRPD
//EKYSCF   DD  DISP=SHR,
//             DSN=&EKYSCF..EKYSCF
//EKYSIDS  DD  DISP=OLD,
//             DSN=&EKYSIDS..EKYSIDS
//SELPRINT DD  SYSOUT=*
//*
//*-------------------------------------------------------------------*
//*  DATASETS REQUIRED BY DPROP MODULES                               *
//*-------------------------------------------------------------------*
//EKYRESLB DD  DISP=SHR,
//             DSN=&EKYPREF..EKYRESLB
//EKYLOG   DD  SYSOUT=*
//*
```

Figure 153. EKYUSC1P: Sample JCL to Execute the SCF Compare Utility

The DD statements //EKYRESLB and //EKYLOG are common to many IMS DPROP components and are described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The remaining statements in Figure 153 are described in the following list.

PROC statement

Defines the various keyword parameters of the JCL. When you call EKYUSC1P, you must provide values for the following keywords. Data set names must be enclosed in single quotes.

EKYPREF The high-level qualifier for the IMS DPROP RESLIB.

EKYGRPD The high-level qualifier for the group definitions file generated by the GUU.

EKYSCF The high-level qualifiers for the Selector control file, which contains the IMS sources for the propagation groups.

EKYSIDS The high-level qualifiers for the sequential SCF control statements file, which contains the SCF control statements generated by the SCF Compare utility.

EXEC Statement
Calls the SCF Compare utility.

//EKYGRPD DD Statement
Describes the group definitions file generated by the GUU.

//EKYSCF DD statement
Describes the Selector control file, which contains the IMS sources for the propagation groups.

//EKYSIDS DD statement
Describes the sequential SCF control statements file, which contains the SCF control statements generated by the SCF Compare utility.

//SELPRINT DD statement
Describes the file the SCF Compare utility uses to write warning and error messages.

EKYUSC1A: Sample JCL to Allocate the SCF control statements File

Figure 154 on page 261 shows the sample JCL for EKYUSC1A, which deletes and allocates the SCF control statements file. EKYUSC1A contains the following steps:

DELETE To delete the SCF control statements file, //EKYSIDS.

ALLOC To allocate the new //EKYSIDS SCF control statements file.

Before you submit the JCL:

- Replace `JJJJJJJ` with valid job card details.
- Replace all occurrences of `1111111` with the high-level qualifiers for the SCF control statements file.

```
//JJJJJJJ  JOB   *** VALID JOB CARD DETAILS ***
//*-------------------------------------------------------------------*
//*     JOB EKYUSC1A TO ALLOCATE THE SCF CONTROL STATEMENTS FILE      *
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//* CHANGE THE FOLLOWING TO COMPLY WITH YOUR INSTALLATION STANDARDS:- *
//* ----------------------------------------------------------------- *
//*                                                                   *
//*  JJJJJJJ   = JOB NAME                                             *
//*  1111111   = SCF Control Statements File HLQ                      *
//*                                                                   *
//*-------------------------------------------------------------------*
//*
//*****************************************************************
//* DELETE THE SCF CONTROL STATEMENTS FILE
//*****************************************************************
//*
//DELETE      EXEC PGM=IEFBR14
//EKYSIDS     DD   DSN=1111111.EKYSIDS,DISP=(MOD,DELETE),
//            UNIT=SYSDA,SPACE=(800,(10,5))
//*
//*****************************************************************
//* ALLOCATE THE SCF CONTROL STATEMENTS FILE
//*****************************************************************
//*
//ALLOC       EXEC PGM=IEFBR14,COND=(4,LT)
//EKYSIDS     DD   DSN=1111111.EKYSIDS,DISP=(NEW,CATLG),UNIT=SYSDA,
//            DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),SPACE=(800,(10,5))
//*
//*-------------------------------------------------------------------*
```

Figure 154. EKYUSC1A: Sample JCL to Allocate the SCF control statements File

SCF Compare Utility Output Messages

SCF Compare utility warning and error messages begin with EKYB7. For more information on the SCF Compare utility messages and codes, see *IMS DPROP Messages and Codes*.

SCF Compare Utility Return Codes

The SCF Compare utility return codes and meanings are:

0 All the groups in the group definition file were processed successfully. The required SCF control statements were generated.

4 Warning messages were issued because there is a propagation group defined in the SCF that is not defined in the group definitions file. If that group is no longer being propagated, delete it from the SCF using the DELGROUP control statement and the SCF Apply utility.

8 One or more of the propagation groups in the group definitions file did not process successfully.

SCF Apply Utility

The SCF Apply utility updates, lists, or updates and lists the propagation group definitions in the SCF, based on the control statements in the SCF control statements file, //EKYSIDS.

When the SCF Apply utility deletes a propagation group from the SCF, it also deletes the records in the uncommitted log record data set relating to that propagation group.

When the SCF Apply utility adds databases, segments, or fields to a propagation group, it uses the DBDs in the DBD library for validation.

The SCF Apply utility performs the following steps:

1. It reads the ADD, DELETE, and LIST control statements from the SCF control statements data set, //EKYSIDS.
2. It processes the control statements based on type.
   - If the control statement is an ADD type, the SCF Apply utility:
     a. Validates the parameters specified on the control statement. The validation it performs depends on the particular ADD control statement it is processing.

        For example, when adding databases, segments, or fields to a group the SCF Apply utility uses the database definitions in the DBD library for validation.
     b. Adds the relevant records to the SCF, if the input in valid.
     c. Writes an audit trail record to the audit trail file, //EKYAUDIT, for each record added to the SCF.
   - If the control statement is a DELETE type, the SCF Apply utility:
     a. Validates the parameters specified on the control statement. The validation it performs depends on the particular DELETE control statement it is processing.

        For example, when it deletes a database from a propagation group, the SCF Apply utility ensures that the propagation group is defined in the SCF.
     b. Deletes the relevant records from the SCF if the input is valid.
     c. Cleans up the uncommitted log record data set by deleting from the data set all records related to propagation groups it deleted from the SCF.
     d. Writes an audit trail record to the audit trail file, //EKYAUDIT, for each record deleted from the SCF.
   - If the control statement is a LIST type, the SCF Apply utility:
     a. Validates the parameters specified on the control statement. The validation it performs depends on the particular LIST control statement it is processing.

        For example, when listing the propagation group details, the SCF Apply utility first ensures that the propagation group is defined in the SCF.
     b. Lists the relevant SCF records in the SCF Listing file (//EKYLIST) if the input is valid.

SCF Apply Utility Input and Output

Figure 155 shows the input to and output from the SCF Apply utility.

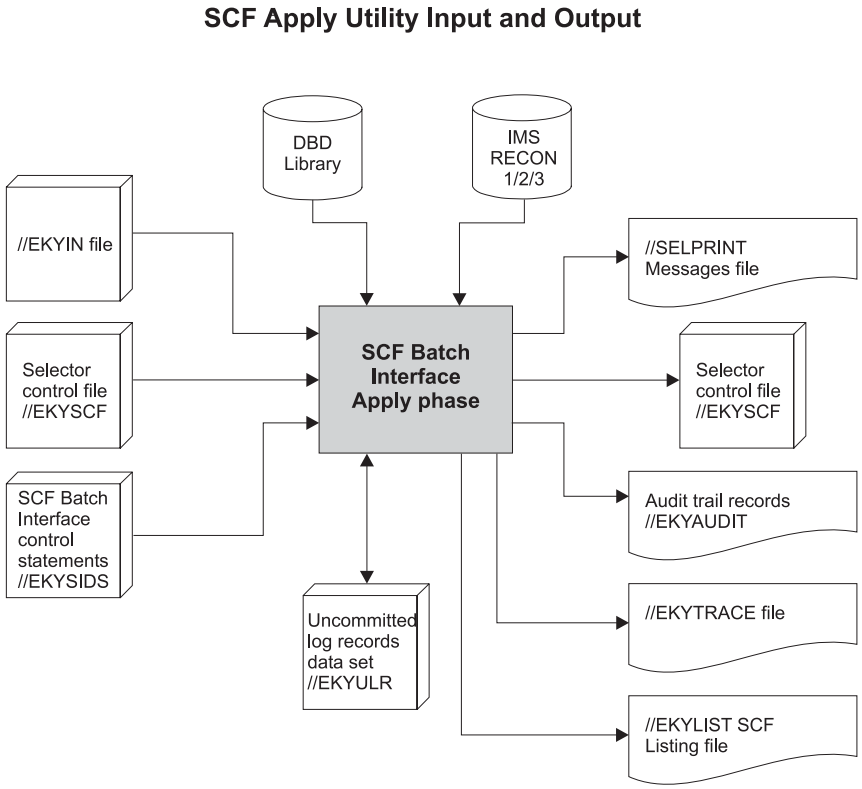

Figure 155. SCF Apply Utility Input and Output

The input to the SCF Apply utility consists of:

- The SCF control statements file, //EKYSIDS, which contains the SCF control statements.

  You can use these control statements from the output of the SCF Compare utility, or you can enter them directly into the EKYSIDS data set. When the control statements are the output of the SCF Compare utility, they bring the SCF definitions of the IMS sources for the propagation groups into line with the IMS DPROP directory definitions.
- The Selector control file, //EKYSCF, which contains the IMS source for the propagation groups as currently defined on the Selector site.
- The uncommitted log records data set, //EKYULR, which contains any uncommitted units of work processed by a previous Selector execution. The Selector collects only units of work that were committed during the time period you specify. If log records collected are neither committed nor abended during this time period, the Selector holds them as uncommitted log records in the ULR data set.
- The IMS DBD Library, //DBDLIB, which contains the IMS database definitions that were processed by the IMS DBDGEN utility. When the SCF Apply utility adds databases, segments, or fields to a propagation group, it uses these definitions for validation.
- The //EKYIN data set, which provides the TRACE control statement to trace the activities of the SCF Apply utility.

- The IMS RECON data sets, //RECON1/2/3, which are used to determine whether the databases are registered to DBRC. The SCF Apply utility cannot add a database to a propagation group unless it is first registered with DBRC.

The output from the SCF Apply utility consists of:

- The Selector control file, //EKYSCF, which contains the updated Selector control information. This information matches the IMS DPROP directory definitions at the Receiver site.
- The ULR data set, //EKYULR, which contains any uncommitted units of work processed by a previous Selector execution.
- The SCF listing file, //EKYLIST, which contains the output from the LIST control statements.
- The audit trail file, //EKYAUDIT, which contains audit trail records that record the SCF Apply utility operations.

  Each time the SCF Apply utility adds a record to the SCF or deletes a record from the SCF, it writes an audit trail record.
- The //EKYTRACE file, which receives IMS DPROP trace output.
- The //SELPRINT file, which contains the print output from the SCF Apply utility.

SCF Apply Utility JCL

This topic describes the following sample JCL:

- EKYUSC2J, the sample JCL to call EKYUSC2P
- EKYUSC2P, the sample JCL to execute the SCF Apply utility

EKYUSC2J: Sample JCL to Call EKYUSC2P

Figure 156 on page 265 shows EKYUSC2J, which calls EKYUSC2P to execute the SCF Apply utility.

Before you submit the JCL:

- Replace `JJJJJJJ` with valid job card details
- Either replace all occurrences of `1111111` with the high-level qualifiers for the required RECON data sets if you are not using dynamic allocation or if you want to override the default RECONs. Otherwise, omit the DD statements.

EKYUSC2J contains only an EXEC statement and, in some cases the RECON statements; therefore, it is not included in the sample JCL library.

```
//jjjjjjj  JOB   ***  Valid job card details ***
//*-------------------------------------------------------------------*
//*       JOB EKYUSC2J TO EXECUTE THE EKYUSC2P PROCEDURE              *
//*                  ( SCF APPLY UTILITY )                            *
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S STANDARDS :    *
//*-------------------------------------------------------------------*
//*                                                                   *
//*  jjjjjjj   - Job name                                             *
//*  1111111   = IMS RECON DATA SETS HLQ                              *
//*              NOTE: ONLY INCLUDE THE IMS RECON DD STATEMENTS IF:-  *
//*              - DYNAMIC ALLOCATION IS NOT IN USE                   *
//*              - THE DEFAULT RECONS ARE TO BE OVERRIDDEN            *
//*                                                                   *
//*-------------------------------------------------------------------*
//*
//RUNSC2P  EXEC PROC=EKYUSC2P
//*
//RECON1     DD   DISP=SHR,
//                DSN=1111111.RECON1,DISP=SHR
//RECON2     DD   DISP=SHR,
//                DSN=1111111.RECON2,DISP=SHR
//RECON3     DD   DISP=SHR,
//                DSN=1111111.RECON3,DISP=SHR
//*
//*-------------------------------------------------------------------*
```

Figure 156. EKYUSC2J: Sample JCL to Call the EKYUSC2P

EKYUSC2P: Sample JCL to Execute the SCF Apply Utility

Figure 157 on page 266 shows the sample JCL, EKYUSCF2, which executes the SCF Apply utility.

Before you invoke the procedure:

- Replace all occurrences of `1111111` with the high-level qualifiers for the IMS DPROP RESLIB.
- Replace all occurrences of `2222222` with the data set name of the IMS DBDLIB.
- Replace all occurrences of `3333333` with the high-level qualifiers for the Selector control file.
- Replace all occurrences of `4444444` with the high-level qualifiers for the uncommitted log record data set.
- Replace all occurrences of `5555555` with the high-level qualifiers for the SCF control statements file.

```
//*-------------------------------------------------------------------*
//*       PROCEDURE EKYUSC2P TO EXECUTE THE SCF APPLY UTILITY         *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//* CHANGE THE FOLLOWING TO COMPLY WITH YOUR INSTALLATION STANDARDS:- *
//*-------------------------------------------------------------------*
//*                                                                   *
//*  1111111   = DPROP RESLIB HIGH LEVEL QUALIFIER (HLQ)              *
//*  2222222   = IMS DBD LIBRARY                                      *
//*  3333333   = SELECTOR CONTROL FILE HLQ                            *
//*  4444444   = UNCOMMITTED LOG RECORDS DATASET HLQ                  *
//*  5555555   = SCF CONTROL STATEMENTS FILE                          *
//*                                                                   *
//*-------------------------------------------------------------------*
//*
//EKYUSC2P PROC EKYPREF=1111111,
//         DBDLIB=2222222,
//         EKYSCF=3333333,
//         EKYULR=4444444,
//         EKYSIDS=5555555
//*
//*-------------------------------------------------------------------*
//RUN      EXEC PGM=EKYB600X
//*
//EKYRESLB DD DISP=SHR,
//            DSN=&EKYPREF..EKYRESLB
//*-------------------------------------------------------------------*
//*  DATASET REQUIRED SPECIFICALLY FOR THE SCF APPLY UTILITY          *
//*-------------------------------------------------------------------*
//DBDLIB   DD DISP=SHR,
//            DSN=&DBDLIB
//EKYSCF   DD DISP=SHR,
//            DSN=&EKYSCF..EKYSCF
//EKYULR   DD DISP=SHR,
//            DSN=&EKYULR..EKYULR
//EKYSIDS  DD DISP=SHR,
//            DSN=&EKYSIDS..EKYSIDS
//*
//EKYLIST  DD SYSOUT=*
//SELPRINT DD SYSOUT=*
```

Figure 157. (Part 2 of 2.).EKYUSC2P: Sample JCL to Execute the SCF Apply Utility (Part 1 of 2)

```
//*-------------------------------------------------------------------*
//*  DATASET REQUIRED BY DBRC                                         *
//*-------------------------------------------------------------------*
//SYSIN    DD DSN=&&SYSIN,UNIT=SYSDA,
//         DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),SPACE=(800,(10,1))
//SYSPRINT DD DSN=&&SYSPRINT,UNIT=SYSDA,
//         DCB=(LRECL=133,BLKSIZE=1330,RECFM=FBA),SPACE=(1300,(10,1))
//*
```

Figure 157. (Part 2 of 2.).EKYUSC2P: Sample JCL to Execute the SCF Apply Utility (Part 2 of 2)

The //EKYRESLB and //EKYAUDIT DD statements are common to many IMS DPROP components and are described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The remaining statements in Figure 157 are described in the following list:

PROC Statement
Defines the various keyword parameters in the JCL. When you call EKYUSC2P, you must provide values for the following keywords. Enclose data set names in single quotes.

EKYPREF The high-level qualifier for the IMS DPROP RESLIB.

DBDLIB The data set name of the IMS DBDLIB.

EKYSCF The high-level qualifier for the Selector control file, which contains the IMS sources for the propagation groups.

EKYULR The high-level qualifiers for the uncommitted log record data set.

EKYSIDS The high level qualifier for the sequential SCF control statements file, which contains the SCF control statements generated by the SCF Compare utility or that you entered manually.

EXEC Statement
Calls the SCF Apply utility.

//DBDLIB DD Statement
Specifies the DBD library containing the IMS database definitions that have been processed by the IMS DBDGEN utility.

//EKYSCF DD Statement
Describes the Selector control file, which contains the IMS sources for the propagation groups.

//EKYULR DD Statement
Describes the uncommitted log record data set, which contains propagation log records that were not yet committed or abended by IMS.

//EKYSIDS DD Statement
Describes the sequential SCF control statements file, which contains the SCF control statements that were generated by the SCF Compare utility or that you entered manually.

//EKYLIST DD Statement
Describes the sequential SCF listing file, which contains the print output from the SCF list control statements.

//SELPRINT DD Statement
Describes the file the SCF Compare utility used to write warning and error messages.

//SYSIN DD Statement
Contains generated input commands to DBRC. This DD statement must be allocated to a writeable data set and ***not*** allocated as instream JCL.

//SYSPRINT DD Statement
Contains the print output from DBRC. You must allocate this DD statement to a readable data set. Do ***not*** allocate it as a SYSOUT file.

SCF control statements

There are two ways you can generate SCF control statements. You can generate them either:

- Automatically with the SCF Compare utility, which generates control statements for differences it finds between the group definitions file and the Selector control file.
- By adding control statements to the data set that you named in the //EKYSIDS DD statement in the SCF Apply utility JCL. See Figure 157 on page 266 for the SCF Apply utility JCL.

These control statements manipulate the contents of records in the SCF. See “SCF Record Layout” on page 307 for descriptions of the SCF record formats.

This topic contains details about each of the SCF control statements, including syntax and examples.

ADDGROUP

Use the ADDGROUP control statement to define a new group in the Selector control file.

The ADDGROUP control statement creates one 0300 (propagation group) record for the group specified on the control statement in the SCF.

The SCF Apply utility validates the parameters on the ADDGROUP control statement by checking that the *group_id* is a valid MVS DDNAME. If the check fails, the SCF Apply utility issues an error message and terminates processing.

If validation is successful, the SCF Apply utility:

- Creates a 0300 record for the propagation group in the SCF. If a 0300 record for the propagation group already exists in the SCF, the SCF Apply utility issues a warning message, but does not overwrite the existing 0300 record.
- Writes an audit trail record to the audit trail table for the 0300 record created.

Each 0300 record in the SCF contains only one propagation group.

Syntax of the ADDGROUP control statement

Figure 158 shows the syntax of the ADDGROUP control statement. Figure 158 on page 269.

```
►►──ADDGROUP GROUP=group_id──;──────────────────────────────────────►◄
```

Figure 158. The ADDGROUP control statement

GROUP= *group_id*
Specifies a propagation group to be defined in a 0300 record in the Selector control file.

The *group_id* must be a valid data definition name. A data definition name consists of 1 to 8 alphanumeric or national characters. The first character must be alphabetic or national.

For more information on naming conventions for data definition names, see *MVS/ESA JCL Reference*.

Example of the ADDGROUP control statement

The example in Figure 159 adds the GROUP01 propagation group to the SCF:

```
ADDGROUP GROUP=GROUP01
```

Figure 159. ADDGROUP control statement Example

ADDDBASE

Use the ADDDBASE control statement to add a database to a propagation group in the SCF.

The ADDDBASE control statement creates the following records in the SCF:

- One 0200 (database) record for each database specified on the control statement
- One 0302 (propagation group/database) record for each database specified on the control statement

The SCF Apply utility validates the parameters on the ADDDBASE control statement by checking that:

- The propagation group is defined in the SCF.
- Each database is defined in the DBD library.
- The database organization of each database is supported by IMS DPROP.
- Each database is registered to DBRC.

If any of the checks fail, the SCF Apply utility issues an error message and terminates processing. If validation is successful, the SCF Apply utility:

- Creates a 0200 record for each database in the SCF. If a 0200 record for any of the databases already exists in the SCF, the SCF Apply utility issues a warning message, but does not overwrite the existing 0200 record.
- Creates a 0302 record in the SCF for each database. If a 0302 record for any of the propagation group/database already exists in the SCF, the SCF Apply utility issues a warning message but does not overwrite the existing 0302 record.
- Writes an audit trail record in the audit trail table for each 0200 record and 0302 record created.

Syntax of the ADDDBASE control statement

Figure 160 shows the syntax of the ADDDBASE control statement.

```
                                  ┌─,──────────┐
                                  ▼            │
►►─ADDDBASE GROUP=group_id,DBD=(──dbase_name──┘)─;──────────────────────►◄
```

Figure 160. The ADDDBASE control statement

GROUP= *group_id*
Specifies the propagation group to which the physical IMS database is to be added.

DBD=(*dbase_name,dbase_name,...* **)**
Specifies the physical IMS databases to be added to the specified propagation group.

Examples of the ADDDBASE control statement

Example 1: The example in Figure 161 adds the IMSCUST database to the GROUP02 propagation group.

```
ADDDBASE GROUP=GROUP02,DBD=(IMSCUST);
```

Figure 161. ADDDBASE control statement Example 1

Example 2: The example in Figure 162 adds the IMSCUST and IMSORD databases to the GROUP03 propagation group.

```
ADDDBASE GROUP=GROUP03,DBD=(IMSCUST,IMSORD);
```

Figure 162. ADDDBASE control statement Example 2

ADDSEGM

Use the ADDSEGM control statement to add a segment to a propagation group in the SCF.

The ADDSEGM control statement creates one 0303 (propagation group/databse/segment) record for each segment in the SCF.

The SCF Apply utility validates the parameters on the ADDSEGM control statement by checking that:
- The propagation group is defined in the SCF.
- The database is defined to the propagation group in the SCF.
- Each segment is defined to the database in the DBD library.

If any of the checks are not valid, the SCF Apply utility issues an error message and terminates processing.

If validation is successful, the SCF Apply utility:

- Creates a 0303 record in the SCF for each segment. If a 0303 record for the propagation group/database/segment already exists in the SCF, the SCF Apply utility issues a warning message but does not overwrite the existing 0303 record.
- Writes an audit trail record in the audit trail table for each 0303 record created.

Syntax of the ADDSEGM control statement

Figure 163 shows the syntax of the ADDSEGM control statement.

```
                                          ┌─,──────────┐
                                          ▼            │
►►─ADDSEGM GROUP=group_id,DBD=dbase_name,SEG=(──seg_name──);──────────────►◄
```

Figure 163. The ADDSEGM control statement

GROUP= *group_id*
Specifies the propagation group to add a segment to.

DBD= *dbase_name*
Specifies the database to add segments to.

SEG=(*seg_name,seg_name...* **)**
Specifies the segments to add.

Examples of ADDSEGM control statement

Example 1: The example in Figure 164 adds the SEG1 segment to the IMSCUST database in the GROUP04 propagation group.

```
ADDSEGM GROUP=GROUP04,DBD=IMSCUST,SEG=(SEG1);
```

Figure 164. ADDSEGM control statement Example 1

Example 2: The example in Figure 165 adds the SEG1 and SEG5 segments to the IMSCUST database in the GROUP05 propagation group.

```
ADDSEGM GROUP=GROUP05,DBD=IMSCUST,SEG=(SEG1,SEG5);
```

Figure 165. ADDSEGM control statement Example 2

ADDFIELD

Use the ADDFIELD control statement to add a field to a group in the SCF.

The ADDFIELD control statement creates one 0304 (propagation group/database/segment/field) record for the field specified on the control statement in the SCF.

The SCF Apply utility validates the parameters on the ADDFIELD control statement by checking that:

- The propagation group is defined in the SCF.
- The database is defined to the propagation group in the SCF.
- The segment is defined to the propagation group in the SCF.

- The field start position is not zero.
- The field length is not zero.
- The field start position is not greater than the segment length.
- The field start position plus the field length is not greater than the segment length.

If any of the checks fail, the SCF Apply utility issues an error message and terminates processing.

If validation is successful, the SCF Apply utility:

- Creates a 0304 record in the SCF for the field. If a 0304 record for the propagation group/database/segment/field already exists in the SCF, the SCF Apply utility issues a warning message but does not overwrite the existing 0304 record.
- Writes an audit trail record in the audit trail table for each 0304 record created.

Syntax of the ADDFIELD control statement

Figure 166 shows the syntax of the ADDFIELD control statement.

```
►►──ADDFIELD GROUP=group_id,DBD=dbase_name,SEG=seg_name,──────────────────────►◄

►►──FLDSTRT=start_byte,FLDLEN=field_length,──┬──────────────────────┬──;──────►◄
                                             └─FLDNAME=field_name───┘
```

Figure 166. The ADDFIELD control statement

GROUP= *group_id*
Specifies the propagation group to add the field to.

DBD= *dbase_name*
Specifies the database to add the field to.

SEG= *seg_name*
Specifies the segment to add the field to.

FLDSTRT= *start_byte*
Specifies the start position within the segment of the field.

FLDLEN= *field_length*
Specifies the length of the field.

FLDNAME= *field_name*
Specifies the name of the field.

Examples of the ADDFIELD control statement

Example: The example in Figure 167 on page 273 adds the FA1 field to the GROUP06/IMSCUST/SEG5 propagation group/database/segment.

```
ADDFIELD  GROUP=GROUP06,DBD=IMSCUST,SEG=SEG5,FLDSTRT=5,FLDLEN=2
          FLDNAME=FA1;
```

Figure 167. ADDFIELD control statement Example

ADDSSID

Use the ADDSSID control statement to define default IMS SSIDs in the SCF or to define IMS SSIDs for a particular propagation group in the SCF.

The ADDSSID control statement creates the following records in the SCF:

- When the GROUP=*group_id* parameter is specified on the control statement, one 0301 group SSID record for each IMS SSID specified on the control statement.
- When the GROUP=*group_id* parameter is not specified on the control statement, one 0101 default SSID record for each IMS SSID specified on the control statement.

Each time you create a propagation group in the SCF, you must specify which IMS subsystems, if any, update the databases in the propagation group. The Selector uses these SSIDs to determine what IMS logs to for a particular propagation group.

During normal operations few IMS subsystems are in use, and these are usually the same for each propagation group. To avoid specifying the same IMS subsystems for each propagation group, you can set up a set of IMS SSIDs in the SCF. IMS SSIDs must be specified for a propagation group only if the set of IMS subsystems that update the databases in a particular propagation group are not the same as the default set of IMS subsystem IDs. IMS SSIDs for a propagation group override the default IMS SSIDs.

If all the databases associated with the propagation group are updated in batch only, do not specify an IMS SSID; instead, use an IMS SSID of NONE (ADDSSID with SSID=NONE). The Selector ignores the set of 0101 default IMS SSID records if there is a 0301 record for a propagation group. If there are 0301 records for a propagation group, they make up the entire set of SSIDs for that propagation group.

If the GROUP=*group_id* parameter is specified on the ADDSSID control statement, the SCF Apply utility ensures that the propagation group is defined in the SCF.

- If the propagation group does not exist, the SCF Apply utility issues an error message and terminates processing.
- If the specified propagation group is located in a 0300 record, the SCF Apply utility:
  - Creates a 0301 record in the SCF for each IMS SSID. If a 0301 record for any propagation group already exists in the SCF, the SCF Apply utility issues a warning message but does not overwrite the existing 0301 record.
  - Writes an audit trail record to the audit trail table for the 0301 record created.

If the *group_id* parameter is not specified, the SCF Apply utility:

- Creates a 0101 record for each IMS SSID. If a 0101 record for an IMS SSID already exists in the SCF, the SCF Apply utility issues a warning message and continues processing. It does not overwrite the existing 0101 record.
- Writes an audit trail record to the audit trail table for the 0101 record created.

Syntax of the ADDSSID control statement

Figure 168 shows the syntax of the ADDSSID control statement.

```
►►─ADDSSID─┬──────────────────┬─,─SSID=─┬─┬─NONE─┬─────────┬─;──────────────►◄
           └─GROUP=group_id─┘         │ │      │         │
                                      │ │      ┌─,─┐     │
                                      │ └─(─────ssid_──)─┘
                                      └─────────────────┘
```

Figure 168. The ADDSSID control statement

GROUP= *group_id*
Specifies the propagation group to define the IMS SSIDs for. Do not specify a value for this parameter if you can use the default set of IMS SSIDs for this propagation group.

SSID=NONE
Specifies that no IMS SSIDs need to be defined for the propagation group because all the databases associated with the propagation group are updated in batch only.

SSID=(*ssid,ssid,...* **)**
Specifies the default IMS SSIDs or propagation groups IDs to add to the SCF. Use this parameter alone if you want to use the default set of SSIDs. Use the GROUP=*group_id* parameter when you want to specify SSIDs other than the default for a propagation group.

Examples of the ADDSSID control statement

Example 1: The example in Figure 169 adds the IMSA IMS SSID to the GROUP07 propagation group.

```
ADDSSID GROUP=GROUP07,SSID=(IMSA)
```

Figure 169. ADDSSID control statement Example 1

Example 2: The example in Figure 170 adds the IMSB and IMSC IMS SSIDs to the `GROUP08` propagation group:

```
ADDSSID GROUP=GROUP08,SSID=(IMSB,IMSC)
```

Figure 170. ADDSSID control statement Example 2

Example 3: The example in Figure 171 adds an SSID of NONE to the GROUP08 propagation group. This definition is required when the databases in the GROUP08 propagation group are updated in batch only.

```
ADDSSID GROUP=GROUP08 ,SSID=NONE;
```

Figure 171. ADDSSID control statement Example 3

Example 4: The example in Figure 172 adds the default IMSA IMS SSID to the SCF.

```
ADDSSID SSID=(IMSA)
```

Figure 172. ADDSSID control statement Example 4

Example 5: The example Figure 173 adds the default IMSA and IMSB IMS SSIDs to the SCF.

```
ADDSSID SSID=(IMSA,IMSB)
```

Figure 173. ADDSSID control statement Example 5

DELGROUP

The DELGROUP control statement deletes the following records from the SCF:

- One 0300 (propagation group) record for each group specified
- One or more 0301 (propagation group SSID) records for each group specified
- One or more 0302 (propagation group/database) records for each group specified
- One or more 0303 (propagation group/databse/segment) records for each group specified
- One or more 0304 (propagation group/database/segment/field) records for each group specified
- One or more 0305 (propagation group/stop time) records for each group specified

The SCF Apply utility checks what is specified on the DELGROUP control statements.

If the ALL parameter is specified, the SCF Apply utility:

- Deletes all 0300, 0301, 0302, 0303, 0304 and 0305 records in the SCF.
- Writes an audit trail record to the audit trail table for each record deleted.

If the GROUP=*group_id* parameter is specified, the SCF Apply utility checks whether the specified propagation groups exist in 0300 records in the SCF.

- If the propagation groups do not exist, the SCF Apply utility issues a warning message and continues processing.
- If the propagation groups exist, the SCF Apply utility:
  - Deletes the 0300, 0301, 0302, 0303, 0304 and 0305 records for the propagation groups specified.
  - Writes an audit trail record to the audit trail table for each record deleted.

The DELGROUP control statement also cleans up the ULR data set by deleting all records in the propagation groups that were deleted from the SCF.

Syntax of the DELGROUP control statement

Figure 174 on page 276 shows the syntax of the DELGROUP control statement.

```
►►──DELGROUP──┬─ALL──────────────────────┬──;──────────────────────────►◄
              │          ┌──,──────┐     │
              │          ▼         │     │
              └─GROUP=(──group_id──┴──)──┘
```

Figure 174. The DELGROUP control statement

ALL
Deletes all propagation groups in the SCF.

GROUP=(*group_id* **)**
Specifies a list of the propagation groups to delete.

Examples of the DELGROUP control statement

Example 1: The example in Figure 175 deletes all propagation groups from the SCF.

```
DELGROUP ALL
```

Figure 175. DELGROUP control statement Example 1

Example 2: The example in Figure 176 deletes only propagation groups GROUP09 and GROUP10 from the SCF.

```
DELGROUP GROUP=(GROUP09,GROUP10)
```

Figure 176. DELGROUP control statement Example 2

DELDBASE

Use the DELDBASE control statement to delete databases or propagation groups and databases from the SCF.

- If you specify the GROUP=*group_id* parameter on the DELDBASE control statement, it deletes:
  - One 0302 (propagation group/database) record for each propagation group/database specified.
  - One or more 0303 (propagation group/databse/segment) records for each propagation group/database specified.
  - One or more 0304 (propagation group/database/segment/field) records for each propagation group/database/segment specified.

When you specify the GROUP=*group_id* parameter on the DELDBASE control statement, the SCF Apply utility checks if the specified propagation group exists in a 0300 record in the SCF.

- If the specified propagation group does not exist, the SCF Apply utility issues an error message and terminates processing.
- If the specified propagation group exists in a 0300 record, the SCF Apply utility checks what parameters are specified on the DELDBASE control statements.
  - If ALL is specified, the SCF Apply utility:

    - Deletes all 0302, 0303, and 0304 records in the SCF that are associated with the specified propagation group.
    - Writes an audit trail record to the audit trail table for each record deleted.
  - If DBD=*dbase_name* is specified, the SCF Apply utility checks to see if the specified propagation groups and databases exist in 0302 records in the SCF.
    - If the propagation groups and databases do not exist, the SCF Apply utility issues a warning message and continues processing.
    - If the propagation groups and databases do exist, the SCF Apply utility:
      - Deletes the 0302, 0303, and 0304 records associated with the propagation groups and databases specified.
      - Writes an audit trail record to the audit trail table for each record deleted.
- If you do not specify the GROUP=*group_id* parameter on the DELDBASE control statement, it deletes:
  - One 0200 (database) record for each database specified.
  - One or more 0202 (database/quiesce timestamp) records for each database specified.

If the *group_id* parameter is *not* specified on the DELDBASE control statement, the SCF Apply utility checks what parameters are specified on the DELDBASE control statements.

- If ALL is specified, the SCF Apply utility checks if any of the databases in the SCF are located in propagation group records in the SCF.
  - If any of the databases are located in propagation group records in the SCF, the SCF issues an error message and terminates processing.
  - If none of the databases are located in propagation group records in the SCF, the SCF:
    - Deletes all 0200 and 0202 records in the SCF.
    - Writes an audit trail record to the audit trail table for each record deleted.
- If you specify the DBD=*dbase_name* parameter on the SCF Apply utility, it checks for a 0200 record in the SCF for each database specified on the database list. If the specified databases are not located in 0200 records, the SCF Apply utility issues an error message and terminates processing.

  The SCF Apply utility then checks if any of the databases are located in propagation group/database records in the SCF.
  - If any databases are located in the propagation group/database records, the SCF Apply utility issues an error message and terminates processing.
  - If none of the databases are located in the propagation group/database records, the SCF Apply utility:
    - Deletes all 0200 and 0202 records in the SCF that are associated with the specified databases on the database list
    - Writes an audit trail record to the audit trail table for each record deleted.

Syntax of the DELDBASE control statement

Figure 177 on page 278 shows the syntax of the DELDBASE control statement.

```
►►─DELDBASE──┬──────────────────┬──,──┬─ALL──────────────────┬──;──────────►◄
             └─GROUP=group_id───┘     │                      │
                                      │        ┌──,──────────┐│
                                      │        ▼             │
                                      └─DBD=(──dbase_name────┘)┘
```

Figure 177. The DELDBASE control statement

GROUP= *group_id*
Specifies the propagation group from which to delete the databases.

If this parameter is not specified, only database records are deleted.

ALL
Deletes all database records or propagation group/database records.

DBD=(*dbase_name,dbase_name,...* **)**
Specifies a list of the database records or propagation group/database records to delete.

Examples of the DELDBASE control statement

Example 1: The example in Figure 178 deletes all databases in the SCF if no 0302 records exist for any of the databases.

```
DELDBASE ALL
```

Figure 178. DELDBASE control statement Example 1

Example 2: The example in Figure 179 deletes databases IMSCUST and IMSORD, if no 0302 records exist for any of the databases:

```
DELDBASE DBD=(IMSCUST,IMSORD)
```

Figure 179. DELDBASE control statement Example 2

Example 3: The example in Figure 180 deletes all databases for the GROUP11 propagation group from the SCF.

```
DELDBASE GROUP=GROUP11,ALL
```

Figure 180. DELDBASE control statement Example 3

Example 4: The example in Figure 181 deletes databases IMSCUST and IMSORD from the GROUP12 propagation group.

```
DELDBASE GROUP=GROUP12,DBD=(IMSCUST,IMSORD)
```

Figure 181. DELEBASE control statement Example 4

DELSEGM

Use the DELSEGM control statement to delete segments for a specified propagation group/database from the SCF.

The DELSEGM control statement deletes the following records from the SCF:

- One 0303 (propagation group/databse/segment) record for each segment specified on the control statement.
- One or more 0304 (propagation group/database/segment/field) records for each segment specified.

The SCF Apply utility:

1. Checks if the specified propagation group exists in a 0300 record in the SCF:
   - If the specified propagation group does not exist, the SCF Apply utility issues an error message and terminates processing.
   - If the specified propagation group is located in a 0300 record, the SCF Apply utility proceeds to Step 2.
2. Checks if the propagation group/database specified on the DELSEGM control statement exists in a 0302 record in the SCF.
   - If it does not exist in a 0302 record the SCF Apply utility issues an error message and terminates processing.
   - If it is located in a 0302 record, the SCF Apply utility:
     - Checks if ALL is specified on the DELSEGM control statement. If so, the SCF Apply utility:
       - Deletes all 0303 records and 0304 records for the specified propagation group/database.
       - Writes an audit trail record to the audit trail table for each record deleted.
   - Checks if SEG=*seg_name* is specified on the DELSEGM control statement. If so, the SCF Apply utility checks that there is a 0303 record in the SCF for each segment specified on the segment list.
     - If the specified segments are not located in 0303 records, the SCF Apply utility issues a warning message and continues processing.
     - If the specified segments are located in 0303 records, the SCF Apply utility:
       - Deletes the 0303 records and 0304 records for the specified propagation group/database from the SCF.
       - Writes an audit trail record to the audit trail table for each record deleted.

Syntax of the DELSEGM control statement

Figure 182 shows the syntax of the DELSEGM control statement.

```
►►──DELSEGM GROUP=group_id,DBD=dbase_name,──┬─ALL──────────────────────┬──;──────────►◄
                                            │        ┌──────────┐      │
                                            │        ▼          │      │
                                            └─SEG=(──seg_name───┴──)───┘
```

Figure 182. The DELSEGM control statement

GROUP= *group_id*
Specifies the propagation group from which to delete.

DBD= *dbase_name*
Specifies the database from which to delete the segments.

ALL
Deletes all segments for the specified propagation group/database.

SEG=(*seg_name,seg_name,...* **)**
Specifies the segments to delete for the specified propagation group/database.

Examples of the DELSEGM control statement

Example 1: The example in Figure 183 deletes all segment records for the GROUP14/IMSCUST propagation group/database:

```
DELSEGM GROUP=GROUP14,DBD=IMSCUST,ALL
```

Figure 183. DELSEGM control statement Example 1

Example 2: The example in Figure 184 deletes the SEG3 and SEG6 segments from the GROUP15/IMSCUST propagation group/database.

```
DELSEGM GROUP=GROUP15,DBD=IMSCUST,SEG=(SEG3,SEG6)
```

Figure 184. DESSEGM control statement Example 2

DELFIELD

Use the DELFIELD control statement to delete fields for the specified propagation group/database/segment from the SCF.

The DELFIELD control statement deletes the 0304 record for the specified field.

The SCF Apply utility:

1. Checks if the specified propagation group exists in a 0300 record in the SCF.
   - If it does not exist in a 0300 record, the SCF Apply utility issues an error message and terminates processing.
   - If the specified propagation group is located in a 0300 record, the SCF Apply utility proceeds to Step 2.
   - Checks if the propagation group/database specified on the DELFIELD control statement exists in a 0302 record in the SCF.
     - If not, the SCF Apply utility issues an error message and terminates processing.
     - If the specified propagation group/database is located in a 0302 record, the SCF Apply utility proceeds to step 3.
2. Checks if the propagation group/database segment specified on the DELFIELD control statement exists in a 0303 record in the SCF.
   - If it does not exist in a 0303 record, the SCF Apply utility issues an error message and terminates processing.
   - If the specified propagation group/database/segment is located in a 0303 record, the SCF Apply utility:

  - Checks if ALL is specified on the DELFIELD control statement. If so, the SCF Apply utility:
    - Deletes all the 0304 records in the SCF for the specified propagation group/database/segment,
    - Writes an audit trail record to the audit trail table.
- Checks if FLDSTRT=*start_byte* is specified on the DELFIELD control statement. If so, the SCF Apply utility checks that the specified propagation group/database/segment/field exists in the SCF.
  - If it does not exist in the SCF, the SCF Apply utility issues a warning message and continues processing.
  - If it does exist, the SCF Apply utility:
    - Deletes the 0304 record for the specified propagation group/database/segment/field.
    - Writes an audit trail record to the audit trail table.

Syntax of the DELFIELD control statement

Figure 185 shows the syntax of the DELFIELD control statement

```
►►──DELFIELD GROUP=group_id,DBD=dbase,SEG=seg_name,──────────────────────────►◄

►►──┬─ALL──────────────────┬──;──────────────────────────────────────────────►◄
    └─FLDSTRT=start_byte───┘
```

Figure 185. The DELFIELD control statement

GROUP= *group_id*
Specifies the propagation group from which to delete the fields.

DBD= *dbase*
Specifies the database from which to delete the fields.

SEG= *seg_name*
Specifies the segment from which to delete the fields.

ALL
Deletes all field records for the specified propagation group/database/segment.

FLDSTRT= *start_byte*
Specifies the field start position of the field to delete.

Examples of the DELFIELD control statement

Example 1: The example in Figure 186 deletes all the fields for the GROUP16/IMSCUST/SEG5 propagation group/database/segment.

```
DELFIELD GROUP=GROUP16,DBD=IMSCUST,SEG=SEG5,ALL
```

Figure 186. DELFIELD control statement Example 1

Example 2: The example in Figure 187 deletes a field with a start position of 5 for the GROUP17/IMSCUST/SEG1 propagation group/database/segment.

```
DELFIELD GROUP=GROUP17,DBD=IMSCUST,SEG=SEG1,FLDSTRT=5
```

Figure 187. DELFIELD control statement Example 2

DELSSID

Use the DELSSID control statement to delete default IMS SSIDs or propagation group/IMS subsystem IDs from the SCF.

The DELSSID control statement deletes the following records from the SCF:

- One 0301 Group/IMS SSID record for each IMS SSID if the GROUP=*group_id* parameter is specified.
- One 0101 default IMS SSID record for each IMS SSID if the GROUP=*group_id* is not specified.

If the GROUP=*group_id* parameter is specified on the DELSSID control statement the SCF Apply utility checks if the specified propagation group exists in a 0300 record in the SCF.

- If the propagation group does not exist, the SCF Apply utility issues an error message and terminates processing.
- If the specified propagation group is located in a 0300 record, the SCF Apply utility checks what is on the DELSSID control statements.
  - If ALL is specified on the DELSSID control statement, the SCF Apply utility:
    - Deletes all 0301 records for the specified propagation group from the SCF,
    - Writes an audit trail record to the audit trail table for each record deleted.
  - If SSID=*ssid* is specified on the DELSSID control statement, the SCF Apply utility checks if the specified propagation group/SSIDs exist in a 0301 record in the SCF.
    - If the propagation group/SSID does not exist, the SCF Apply utility issues a warning message and continues processing.
    - If the propagation group/SSID does exist, the SCF Apply utility:
      - Deletes the 0301 record from the SCF.
      - Writes an audit trail record to the audit trail table for each record deleted.

If the GROUP=*group_id* parameter is *not* specified on the DELDSSID control statement, the SCF Apply utility checks what parameters are specified on the DELSSID control statement.

- If ALL is specified, the SCF Apply utility:
  - Deletes all 0101 records in the SCF.
  - Writes an audit trail record to the audit trail table for each record deleted.
- If SSID=*ssid* is specified on the DELSSID control statement, the SCF Apply utility checks if there is a 0101 record in the SCF for each default SSID specified on the SSID list.
  - If the default SSIDs are not in 0101 records, the SCF Apply utility issues a warning message and continues processing.
  - If the default SSIDs are in the 0101 records, the SCF Apply utility:

- Deletes the 0101 record from the SCF.
- Writes an audit trail record to the audit trail table for each record deleted.

Syntax of the DELSSID control statement

Figure 188 shows the syntax of the DELSSID control statement.

```
►►──DELSSID──┬──────────────────┬──,──┬─ALL──────────────────┬──;──────────►◄
             └─GROUP=group_id───┘     │                       │
                                      └─SSID=──┬─NONE─┬────────┤
                                               │      │        │
                                               └─(──┬─ssid─┬─)─┘
                                                    ▲ ,    │
                                                    └──────┘
```

Figure 188. The DELSSID control statement

GROUP= *group_id*
Specifies the propagation group from which to delete the propagation group/SSID records.

ALL
Deletes all default SSID records or all propagation group/SSID records for the specified propagation group.

SSID=NONE
Deletes SSIDs with a value of NONE. This record is created for a propagation group that is updated in batch only, to suppress the use of the default SSIDs.

SSID=(*ssid,ssid...* **)**
Specifies the default SSIDs or propagation group/SSIDs to delete from the SCF.

Examples of the DELSSID control statement

Example 1: The example in Figure 189 deletes all IMS SSIDs for the GROUP07 propagation group:

```
DELSSID GROUP=GROUP07,ALL
```

Figure 189. DELSSID control statement Example 1

Example 2: The example in Figure 190 deletes the IMSB and IMSC IMS SSIDs from the GROUP08 propagation group.

```
DELSSID GROUP=GROUP08,SSID=(IMSB,IMSC)
```

Figure 190. DELSSID control statement Example 2

Example 3: The example in Figure 191 on page 284 deletes the SSID of NONE from the GROUP08 propagation group:

```
DELSSID GROUP=GROUP08 ,SSID=NONE;
```

Figure 191. DELSSID control statement Example 3

Example 4: The example in Figure 192 deletes all the default IMS SSIDs from the SCF:

```
DELSSID ALL
```

Figure 192. DELSSID control statement Example 4

Example 5: The example in Figure 193 deletes the default IMSA and IMSB IMS SSIDs from the SCF:

```
DELSSID SSID=(IMSA,IMSB)
```

Figure 193. DELSSID control statement Example 5

LISTGROUP

Use the LISTGROUP control statement to write propagation group details to the SCF listing file.

The SCF Apply utility reads information for the groups specified from the following SCF records, formats the information, and writes it to the SCF listing file:
- 0300 (propagation group) record
- 0302 (propagation group/database) records
- 0303 (propagation group/databse/segment) records
- 0304 (propagation group/database/segment/field) records
- 0305 (propagation group/stop time) records

The SCF Apply utility validates the parameters on the LISTGROUP control statement by checking that each of the groups in any list of groups specified is defined by the SCF.

If any of the checks fail, the SCF Apply utility issues a warning message and continues processing.

Syntax of the LISTGROUP control statement

Figure 194 shows the syntax of the LISTGROUP control statement.

```
                          ┌─,──────────┐
                          ▼            │
►►──LISTGROUP──┬─GROUP=(──────group_id──┴──)──┬─;──────────────────────────────►◄
               └─ALL──────────────────────────┘
```

Figure 194. The LISTGROUP control statement

ALL
Specifies that the details of all the propagation groups in SCF are to be listed in the SCF listing file.

GROUP=(group_id,group_id,....)
Specifies the propagation groups whose details are to be listed in the SCF listing file.

Examples of the LISTGROUP control statement

Example 1: The example in Figure 195 lists details for all the propagation groups defined in the SCF in the SCF listing file.

```
LISTGROUP ALL
```

Figure 195. LISTGROUP control statement Example 1

Example 2: The example in Figure 196 lists details for GROUP02 and GROUP03 in the SCF listing file.

```
LISTGROUP GROUP=(GROUP02,GROUP03)
```

Figure 196. LISTGROUP control statement Example 2

Sample Output from the LISTGROUP control statement

Figure 197 on page 286 shows the output from the LISTGROUP control statement.

```
************************************************************************
************************************************************************

   LISTGROUP GROUP=(GROUP02,GROUP03) ;

************************************************************************
************************************************************************

****************************
*   GROUP  :  GROUP02    *
****************************

Data Source Type        : IMS310

Group Stop Timestamp    : 1994-03-07-14.49.59.600000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000001
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              1V00
Record Update Timestamp : 1994-03-07-14.49.59.690000


Group Stop Timestamp    : 1994-03-11-16.28.08.800000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000002
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              3V00
Record Update Timestamp : 1994-03-11-16.28.08.850000


Group Stop Timestamp    : 1994-03-11-16.37.10.500000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000003
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              4V00
Record Update Timestamp : 1994-03-11-16.37.10.590000


Group Stop Timestamp    : 1994-03-11-16.51.33.300000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000004
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              5V00
Record Update Timestamp : 1994-03-11-16.51.33.380000
```

Figure 197. Sample Output from the LISTGROUP control statement (Part 1 of 5)

```
Group Stop Timestamp    : 1994-03-16-11.25.40.513780
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000005
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000               6V00
Record Update Timestamp : 1994-03-16-11.42.21.830000


  Database   :   IMSCUST
  ========================

  Group Database Start Timestamp    : 1994-03-16-10.00.00.000000
  Group Database Start Timestamp ID :
  New Start Timestamp Set           : Y
  Record Update Timestamp           : 1994-03-16-13.59.51.230000


    Segment   :   SEG1
    **********************


    Field Start     Field Length     Field Name

       00007           00002         FSEQNBR1


       00015           00020         FGENERAL


    Segment   :   SEG2
    **********************


    Field Start     Field Length     Field Name

       00015           00020         FGENERAL
```

Figure 197. Sample Output from the LISTGROUP control statement (Part 2 of 5)

```
  Segment   :   SEG3
  **********************

  Segment   :   SEG4
  **********************


  Field Start     Field Length     Field Name

     00007           00002         FSEQNBR1


     00015           00020         FGENERAL


     00035           00001         FSBYT1


     00036           00001         FSBYT2


     00190           00010         FZONE42


  Segment   :   SEG5
  **********************


Database   :   IMSORD
=======================

Group Database Start Timestamp    : 1994-03-16-10.00.00.000000
Group Database Start Timestamp ID :
New Start Timestamp Set           : Y
Record Update Timestamp           : 1994-03-16-13.59.51.230000


  Segment   :   SEG1
  **********************


  Field Start     Field Length     Field Name

     00010           00002         FSEQNBR1


     00020           00020         FGENERAL
```

Figure 197. Sample Output from the LISTGROUP control statement (Part 3 of 5)

```
    Segment   :   SEG2
    **********************


    Field Start     Field Length     Field Name

       00001           00020         FGENERAL


    Segment   :   SEG3
    **********************

    Segment   :   SEG4
    **********************


    Field Start     Field Length     Field Name

       00001           00002         FSEQNBR1


       00010           00020         FGENERAL


       00035           00001         FSBYT1


    Segment   :   SEG5
    **********************

****************************
*   GROUP  :  GROUP03    *
****************************

Data Source Type        : IMS310

Group Stop Timestamp    : 1994-03-10-15.45.42.368319
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000009
PRDS Data Set Name      : DPROPTT.TT3VZ027.GROUP03.G002              7V00
Record Update Timestamp : 1994-03-14-16.53.51.320000


Group Stop Timestamp    : 1994-03-16-09.00.00.000000
Group Stop Timestamp ID :
Record Created by       : TSMF
Group Stop Selected     : N
PRDS Sequence Number    : 000000000
PRDS Data Set Name      :
Record Update Timestamp :
```

Figure 197. Sample Output from the LISTGROUP control statement (Part 4 of 5)

```
Database   :   TDBDPH
=======================

Group Database Start Timestamp    : 1994-03-07-09.00.00.000000
Group Database Start Timestamp ID :
New Start Timestamp Set           : N
Record Update Timestamp           : 1994-03-14-16.53.51.320000


  Segment   :   SH1000
  **********************


  Field Start     Field Length     Field Name

     00007           00002         FSEQNBR1


     00015           00020         FGENERAL


  Segment   :   SH1300
  **********************


  Field Start     Field Length     Field Name

     00015           00020         FGENERAL
```

Figure 197. Sample Output from the LISTGROUP control statement (Part 5 of 5)

LISTDBASE

Use the LISTDBASE control statement to write propagation group and propagation group/database details or database details to the SCF listing file.

When the GROUP=*group_id* parameter is specified, LISTDBASE writes propagation group details and propagation group/database details to the SCF listing file. The SCF Apply utility:

- Reads information for the group specified in the following SCF records, and formats the information, and writes it to the SCF listing file:
  - 0300 (propagation group) records
  - 0305 (propagation group/stop time) records
- Reads information for each of the propagation group/databases specified in the following SCF records, and formats and writes the information to the SCF listing file:
  - 0302 (propagation group/database) record
  - 0303 (propagation group/databse/segment) records
  - 0304 (propagation group/database/segment/field) records
- Validates the parameters on the LISTDBASE control statement by checking that:
  - The propagation group specified is defined in the SCF.
  - Each of the databases in any list of databases specified is defined to the specified group in the SCF.

If any of the checks fail, the SCF Apply utility issues a warning message and continues processing.

When the GROUP=*group_id* keyword is omitted, the database details for the databases specified are written to the SCF listing file:

- Reads information for each of the databases specified in the following SCF records, and formats the information and writes it to the SCF listing file:
  - 0200 (database) record
  - 0202 (database/quiesce timestamp) records
- Validates the parameters on the LISTDBASE control statement by checking that each of the databases in any list of databases specified is defined in the SCF.

If any of the checks fail, the SCF Apply utility issues a warning message and continues processing.

Figure 198 shows the syntax of the LISTDBASE control statement.

```
►►──LISTDBASE──┬──────────────────┬──,──┬──ALL──────────────────────┬──;──────►◄
               └─GROUP=group_id───┘     │                           │
                                        │          ┌──────────────┐  │
                                        │          ▼              │  │
                                        └─DBD=──(──┬─dbase_name──┬─┴─)──┘
```

Figure 198. The LISTDBASE control statement

GROUP= *group_id*
Specifies the propagation group whose details are written to the SCF listing file.

ALL
Writes either all the propagation group/database details for the group specified or all the database details to the SCF listing file.

DBD= *(dbase_name,dbase_name,....)*
Specifies the databases whose database details or propagation group/database details are written to the SCF listing file.

Examples of the LISTDBASE control statement

Example 1: The example in Figure 199 lists the databases defined in the SCF.

```
LISTDBASE ALL
```

Figure 199. LISTDBASE control statement Example 1

Example 2: The example in Figure 200 lists database details for databases IMSCUST and IMSORD in the SCF listing file.

```
LISTDBASE DBD=(IMSCUST,IMSORD)
```

Figure 200. LISTDBASE control statement Example 2

Example 3: The example in Figure 201 lists the propagation group details for GROUP02 and the propagation group/database details for all databases in GROUP02 in the SCF listing file.

```
LISTDBASE GROUP=GROUP02, ALL
```

Figure 201. LISTDBASE control statement Example 3

Example 4: The example in Figure 202 lists the group details for GROUP02 and the propagation group/database details for GROUP02/ /IMSCUST and GROUP02/IMSORD in the SCF listing file.

```
LISTDBASE  GROUP=Group02, DBD=(IMSCUST,IMSORD)
```

Figure 202. LISTDBASE control statement Example 4

Sample Output from the LISTDBASE control statement

Figure 203 on page 293 and Figure 204 on page 296 show LISTDBASE control statement output.

```
************************************************************************
************************************************************************

   LISTDBASE GROUP=GROUP02, DBD=IMSCUST ;

************************************************************************
************************************************************************

****************************
*   GROUP  :  GROUP02    *
****************************

Data Source Type        : IMS310

Group Stop Timestamp    : 1994-03-07-14.49.59.600000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000001
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000               1V00
Record Update Timestamp : 1994-03-07-14.49.59.690000


Group Stop Timestamp    : 1994-03-11-16.28.08.800000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000002
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000               3V00
Record Update Timestamp : 1994-03-11-16.28.08.850000


Group Stop Timestamp    : 1994-03-11-16.37.10.500000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000003
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000               4V00
Record Update Timestamp : 1994-03-11-16.37.10.590000


Group Stop Timestamp    : 1994-03-11-16.51.33.300000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000004
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000               5V00
Record Update Timestamp : 1994-03-11-16.51.33.380000
```

Figure 203. Sample Output from the LISTDBASE control statement (Part 1 of 3)

```
Group Stop Timestamp    : 1994-03-16-11.25.40.513780
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000005
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000               6V00
Record Update Timestamp : 1994-03-16-11.42.21.830000


  Database   :   IMSCUST
  =======================

  Group Database Start Timestamp    : 1994-03-16-10.00.00.000000
  Group Database Start Timestamp ID :
  New Start Timestamp Set           : Y
  Record Update Timestamp           : 1994-03-16-13.59.51.230000


    Segment   :   SEG1
    **********************


    Field Start     Field Length     Field Name

       00007           00002         FSEQNBR1


       00015           00020         FGENERAL
```

Figure 203. Sample Output from the LISTDBASE control statement (Part 2 of 3)

```
Segment   :   SEG2
**********************


Field Start     Field Length     Field Name

   00015           00020         FGENERAL


Segment   :   SEG3
**********************

Segment   :   SEG4
**********************


Field Start     Field Length     Field Name

   00007           00002         FSEQNBR1


   00015           00020         FGENERAL


   00035           00001         FSBYT1


   00036           00001         FSBYT2


   00190           00010         FZONE42


Segment   :   SEG5
**********************
```

Figure 203. Sample Output from the LISTDBASE control statement (Part 3 of 3)

```
SAMPLE OUTPUT FROM THE LISTDBASE CONTROL STATEMENT :


*******************************************************************************
*******************************************************************************

   LISTDBASE DBD=IMSCUST ;

***********************************************************************
***********************************************************************

******************************
*   DATABASE  :  IMSCUST    *
******************************

  Database Quiesce Timestamp    : 1994-03-07-13.28.43.662404
  Database Quiesce Timestamp ID : TSM01


  Database Quiesce Timestamp    : 1994-03-07-14.03.46.617765
  Database Quiesce Timestamp ID : TSM02


  Database Quiesce Timestamp    : 1994-03-11-16.15.35.269299
  Database Quiesce Timestamp ID : TSM07
```

Figure 204. Sample Output from the LISTDBASE control statement

LISTSEGM

Use the LISTSEGM control statement to write propagation group, propagation group/database, and propagation group/database/segment details to the SCF listing file. The SCF Apply utility:

- Reads information for the group specified from the following SCF records, and formats and writes the information to the SCF listing file:
  - 0300 (propagation group) record
  - 0305 (propagation group/stop time) records
- Reads information for the propagation group/database specified in the 0302 (propagation group/database) record and formats and writes the information to the SCF listing file.
- Reads information for the propagation group/database/segments specified from the following SCF records, and formats and writes the information to the SCF listing file:
  - 0303 (propagation group/databse/segment) record
  - 0304 (propagation group/database/segment/field) records
- Validates the parameters on the LISTSEGM control statement by checking that:
  - The propagation group specified is defined in the SCF.
  - The propagation group/database specified is defined in the SCF.
  - Each of the segments in any list of segments specified is defined to the specified group/database in the SCF.

  If any of the checks fail, the SCF Apply utility issues a warning message and continues processing.

Syntax of the LISTSEGM control statement

Figure 205 shows the syntax of the LISTSEGM control statement.

```
►►──LISTSEGM GROUP=group_id,DBD=dbase_name,──┬─ALL──────────────────┬──;──────────►◄
                                             │        ┌──────┐      │
                                             │        ▼      ,      │
                                             └─SEG=(──seg_name──)───┘
```

Figure 205. The LISTSEGM control statement

GROUP= *group_id*
Specifies the propagation group whose details are written to the SCF listing file.

DBD= *dbase_name*
Specifies the database whose propagation group/database details are written to the SCF listing file.

ALL
Writes all the propagation group/database/segment details for the group/database specified to the SCF listing file.

SEG= *(seg_name,seg_name,....)*
Specifies the segments whose propagation group/database/segment details are written to the SCF listing file.

Examples of the LISTSEGM control statement

Example 1: The example in Figure 206 lists the following details in the SCF listing file.

- Propagation group details for GROUP02
- Propagation group/database details for GROUP02/IMSCUST
- Propagation group/database/segment details for all segments in GROUP02/IMSCUST

```
LISTSEGM  GROUP=GROUP02, DBD=IMSCUST, ALL
```

Figure 206. LISTSEGM control statement Example 1

Example 2: The example in Figure 207 lists the following details in the SCF Listing file.

- Propagation group details for GROUP02
- Propagation group/database details for GROUP02/IMSCUST
- Propagation group/database/segment for segments SEG1 and SEG5 in GROUP02/IMSCUST

```
LISTSEGM  GROUP=GROUP02, DBD=IMSCUST, SEG=(SEG1,SEG5)
```

Figure 207. LISTSEGM control statement Example 2

Sample Output from the LISTSEGM control statement:

Figure 208 shows LISTSEGM control statement output.

```
*************************************************************************
*************************************************************************

   LISTSEGM GROUP=GROUP02, DBD=IMSCUST, SEG=(SEG1,SEG5) ;

*************************************************************************
*************************************************************************

****************************
*   GROUP  :  GROUP02    *
****************************

Data Source Type        : IMS310

Group Stop Timestamp    : 1994-03-07-14.49.59.600000
Group Stop Timestamp Id :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000001
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              1V00
Record Update Timestamp : 1994-03-07-14.49.59.690000


Group Stop Timestamp    : 1994-03-11-16.28.08.800000
Group Stop Timestamp Id :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000002
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              3V00
Record Update Timestamp : 1994-03-11-16.28.08.850000


Group Stop Timestamp    : 1994-03-11-16.37.10.500000
Group Stop Timestamp Id :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000003
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              4V00
Record Update Timestamp : 1994-03-11-16.37.10.590000


Group Stop Timestamp    : 1994-03-11-16.51.33.300000
Group Stop Timestamp Id :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000004
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              5V00
Record Update Timestamp : 1994-03-11-16.51.33.380000
```

Figure 208. Sample Output from the LISTSEGM control statement (Part 1 of 2)

```
Group Stop Timestamp    : 1994-03-16-11.25.40.513780
Group Stop Timestamp Id :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000005
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000               6V00
Record Update Timestamp : 1994-03-16-11.42.21.830000


  Database   :   IMSCUST
  ========================

  Group Database Start Timestamp    : 1994-03-16-10.00.00.000000
  Group Database Start Timestamp Id :
  New Start Timestamp Set           : Y
  Record Update Timestamp           : 1994-03-16-13.59.51.230000


    Segment   :   SEG1
    **********************


    Field Start     Field Length     Field Name

       00007           00002         FSEQNBR1


       00015           00020         FGENERAL


    Segment   :   SEG5
    **********************
```

Figure 208. Sample Output from the LISTSEGM control statement (Part 2 of 2)

LISTFIELD

Use the LISTFIELD control statement to write propagation group, propagation group/database, propagation group/database/segment and propagation group/database/segment/field details to the SCF listing file. The SCF Apply utility:

- Reads information for the group specified in the following SCF records, and formats and writes the information to the SCF listing file:
  - 0300 (propagation group) record
  - 0305 (propagation group/stop time) records
- Reads information for the propagation group/database specified in the 0302 (propagation group/database) record record, and formats and writes the information to the SCF listing file
- Reads information for the propagation group/database/segment specified from the 0303 (propagation group/databse/segment) record, and formats and writes the information to the SCF listing file
- Reads information for the propagation group/database/segment/fields specified in the 0304 (propagation group/database/segment/field) records, and formats and writes the information to the SCF listing file
- Validates the parameters on the LISTFIELD control statement by checking that:
  - The propagation group specified is defined in the SCF.
  - The propagation group/database specified is defined in the SCF.

– The propagation group/database/segment specified is defined in the SCF.
– Each of the fields in any list of fields specified is defined to the specified group/database/segment in the SCF.

If any of the checks fail, the SCF Apply utility issues a warning message and continues processing.

Syntax of the LISTFIELD control statement

Figure 209 shows the syntax of the LISTFIELD control statement.

```
►►──LISTFIELD GROUP=group_id,DBD=dbase_name,SEG=seg_name,──────────────────────►◄

►►──┬─ALL──────────────────────────┬──;──────────────────────────────────────►◄
    │              ┌──────────┐    │
    │              ▼          ,    │
    └─FLDSTRT=(──start_byte──┘──)──┘
```

Figure 209. The LISTFIELD control statement

GROUP= *group_id*
Specifies the propagation group whose details are written to the SCF listing file.

DBD= *dbase_name*
Specifies the database whose propagation group/database details are written to the SCF listing file.

SEG=seg_name
Specifies the segment whose propagation group/database/segment details are to be written to the SCF listing file.

ALL
Writes all the propagation group/database/segment/ field details for the group/database/segment specified to the SCF Listing file.

FLDSTRT= *(start_byte,start_byte,....)*
Specifies the start position of the fields whose propagation group/database/segment/field details are to be written to the SCF listing file. Fields are identified by their start position.

Examples of the LISTFIELD control statement

Example 1: The example in Figure 210 lists the following details in the SCF listing file:

- Propagation group details for GROUP02
- Propagation group/database details for GROUP02/IMSCUST
- Propagation group/database/segment details for SEG1
- Propagation group/database/segment/field details for all fields in GROUP02/IMSCUST/SEG1

```
LISTFIELD GROUP=GROUP02, DBD=IMSCUST, SEG=SEG1, ALL
```

Figure 210. LISTFIELD control statement Example 1

Example 2: The example in Figure 211 lists the following details in the SCF listing file:

- Propagation group details for GROUP02
- Propagation group/database details for GROUP02/IMSCUST
- Propagation group/database/segment details for SEG1
- Propagation group/database/segment/field details for the fields starting at positions 7 and 15 in GROUP02/IMSCUST/SEG1

```
LISTFIELD GROUP=GROUP02, DBD=IMSCUST, SEG=SEG1, FLDSTRT=(7,15)
```

Figure 211. LISTFIELD control statement Example 2

Sample Output from the LISTFIELD control statement

Figure 212 on page 302 shows LISTFIELD control statement output.

```
***************************************************************************
***************************************************************************

   LISTFIELD GROUP=GROUP02, DBD=IMSCUST, SEG=SEG1, FLDSTRT=(7,15) ;

***************************************************************************
***************************************************************************

****************************
*   GROUP  :  GROUP02    *
****************************

Data Source Type        : IMS310

Group Stop Timestamp    : 1994-03-07-14.49.59.600000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000001
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              1V00
Record Update Timestamp : 1994-03-07-14.49.59.690000


Group Stop Timestamp    : 1994-03-11-16.28.08.800000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000002
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              3V00
Record Update Timestamp : 1994-03-11-16.28.08.850000


Group Stop Timestamp    : 1994-03-11-16.37.10.500000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000003
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              4V00
Record Update Timestamp : 1994-03-11-16.37.10.590000


Group Stop Timestamp    : 1994-03-11-16.51.33.300000
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000004
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000              5V00
Record Update Timestamp : 1994-03-11-16.51.33.380000
```

Figure 212. Sample Output from the LISTFIELD control statement (Part 1 of 2)

```
Group Stop Timestamp    : 1994-03-16-11.25.40.513780
Group Stop Timestamp ID :
Record Created by       : Selector
Group Stop Selected     : Y
PRDS Sequence Number    : 000000005
PRDS Data Set Name      : DPROPTT.TT3ZZ011.GROUP02.G000               6V00
Record Update Timestamp : 1994-03-16-11.42.21.830000


  Database   :   IMSCUST
  ========================

  Group Database Start Timestamp    : 1994-03-16-10.00.00.000000
  Group Database Start Timestamp Id :
  New Start Timestamp Set           : Y
  Record Update Timestamp           : 1994-03-16-13.59.51.230000


    Segment   :   SEG1
    **********************

       00007           00002         FSEQNBR1


       00015           00020         FGENERAL
```

Figure 212. Sample Output from the LISTFIELD control statement (Part 2 of 2)

LISTSSID

Use the LISTSSID control statement to write the default IMS SSIDs defined in the SCF or to write the IMS subsystem IDs associated with propagation groups to the SCF listing file.

When the GROUP=*group_id* keyword is specified, the IMS SSIDs associated with the propagation groups specified are written to the SCF listing file. The SCF Apply utility:

- Reads, formats, and writes the 0301 (propagation group SSID) records to the SCF listing file if IMS SSIDs are assigned to the group.
- Reads, formats, and writes the 0101 (default SSID) records to the SCF listing file if IMS SSIDs are not assigned to the group.
- Validates the parameters on the LISTSSID control statement by checking that the propagation groups specified are defined in the SCF. If a group is not defined, the SCF Apply utility issues a warning message and continues processing.

When the ALL parameter is specified, the IMS SSIDs associated with all the propagation groups in the SCF are written to the SCF listing file. The SCF Apply utility:

- Reads, formats, and writes the 0301 propagation group SSID records to the SCF listing file if IMS SSIDs have been assigned to a group.
- Reads, formats, and writes the 0101 (default SSID) records to the SCF listing file if IMS SSIDs have not been assigned to a group.

When the DEFAULT parameter is specified, the default IMS SSIDs are written to the SCF listing file. The SCF Apply utility reads, formats, and writes the 0101 (default SSID) records to the SCF listing file.

Syntax of the LISTSSID control statement

Figure 213 shows the syntax of the LISTSSID control statement.

```
►►─LISTSSID──┬─ALL──────────────────┬──;──────────────────────────►◄
             │─DEFAULT──────────────│
             │          ┌─,──────┐  │
             │          ▼        │  │
             └─GROUP=(──group_id──┘──)─┘
```

Figure 213. The LISTSSID control statement

GROUP= *group_id*
Specifies the propagation groups whose associated IMS SSIDs are listed in the SCF listing file.

ALL
Lists the IMS SSIDs associated with each of the groups defined in the SCF in the SCF listing file.

DEFAULT
Lists the default IMS SSIDs defined in the SCF in the SCF listing file.

Examples of the LISTSSID control statement

Example 1: The example in Figure 214 lists the IMS SSIDs for all propagation groups defined in the SCF in the SCF listing file.

```
LISTSSID ALL
```

Figure 214. LISTSSID control statement Example 1

Example 2: The example in Figure 215 lists the IMS SSIDs for propagation groups GROUP02 and GROUP03 in the SCF listing file.

```
LISTSSID GROUP=(GROUP02,GROUP03)
```

Figure 215. LISTSSID control statement Example 2

Example 3: The example in Figure 216 lists the default IMS SSIDs in the SCF listing file.

```
LISTSSID DEFAULT
```

Figure 216. LISTSSID control statement Example 3

Sample Output from the LISTSSID control statement:

Figure 217 on page 305 and Figure 218 on page 305 show LISTSSID control statement output.

```
*************************************************************************
*************************************************************************

   LISTSSID GROUP=(GROUP02,GROUP03) ;

*************************************************************************
*************************************************************************

The following IMS Subsystem IDs are assigned to Group GROUP02   :

        IMSA
        IMSB


Group GROUP03  uses the default set of IMS Subsystem IDs
The following default IMS Subsystem Ids have been defined  :

        IMSC
        IMSE
```

Figure 217. Sample Output from the LISTSSID control statement

```
SAMPLE OUTPUT FROM THE LISTSSID CONTROL STATEMENT :


*************************************************************************
*************************************************************************

   LISTSSID DEFAULT ;

*************************************************************************
*************************************************************************

The following default IMS Subsystem IDs have been defined  :

        IMSC
        IMSE
```

Figure 218. Sample Output from the LISTSSID control statement

LISTALL GROUPS

Use the LISTALL GROUPS control statement to write a list of the propagation groups defined in the SCF to the SCF listing file.

The SCF Apply utility reads, formats, and writes the list of propagation groups from the 0300 (propagation group) records to the SCF listing file.

Syntax of the LISTALL GROUPS control statement

Figure 219 on page 306 shows the syntax of the LISTALL GROUPS control statement.

```
►►──LISTALL GROUPS──;──────────────────────────────────────────────────►◄
```

Figure 219. The LISTALL GROUPS control statement

GROUPS
Lists in the SCF Listing file the group names of all the propagation groups that exist in the SCF.

Example of the LISTALL GROUPS control statement

Example: The example in Figure 220 lists the group names of all the propagation groups defined in the SCF.

```
LISTALL GROUPS
```

Figure 220. LISTALL GROUPS control statement Example

Sample Output from the LISTALL GROUPS control statement

Figure 221 shows LISTALL group control statement output.

```
***************************************************************************
***************************************************************************

   LISTALL GROUPS ;

***************************************************************************
***************************************************************************

The following Groups are defined in the Selector Control File  :

        GROUP01
        GROUP02
        GROUP03
        GROUP04
```

Figure 221. Sample Output from the LISTALL GROUPS control statement

SCF Apply Utility Output Messages

Warning and error messages for the SCF Apply utility begin with EKYB6. See*IMS DPROP Messages and Codes* for descriptions of error messages.

SCF Apply Utility Return Codes

The SCF Apply utility provides the following return codes:

0 All the control statements in the SCF control statements file (//EKYSIDS) processed successfully.

4 Warning messages were issued for one of the following reasons:
- You tried to add a record to the SCF that already exists.
- You tried to list or delete a record that does not exist in the SCF.
- No groups are defined in the SCF.

8 One or more of the control statements in the SCF control statements file (//EKYSIDS) were not processed successfully.

SCF Record Layout

Figure 222 on page 308 shows the layout of the Selector control file records and the relationships between them. Figure 223 on page 309 shows the format of the records.

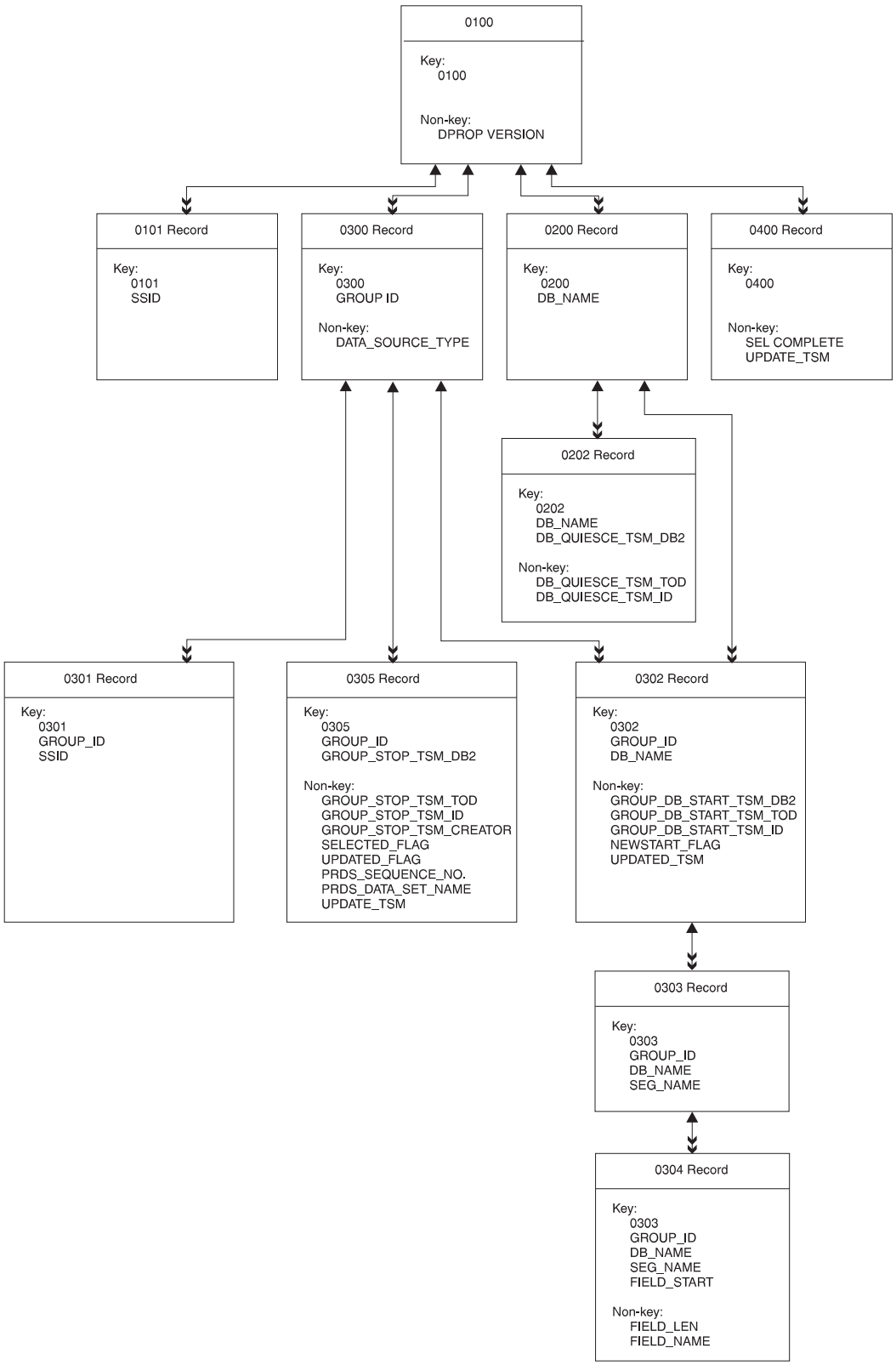

Figure 222. Layout of the Selector Control File Records

Format of Selector Control File Records

Record Type	Key Part 2	Key Part 3	Key Part 4	Key Part 5	Data Portion of Record						
0100					IMS DPROP version	IMS DPROP release	IMS DPROP modification				
0101	SSID										
0200	DB name										
0202	DB name	DB quiesce TSM (DB2)			DB quiesce TSM (TOD)	DB quiesce TSM ID					
0300	Group ID				DATA source type						
0301	Group ID	SSID									
0302	Group ID	DB name			Group/DB start TSM (DB2)	Group/DB start TSM (TOD)	Group/DB start TSM ID	Newstart flag	Update TSM		
0303	Group ID	DB name	Segment name								
0304	Group ID	DB name	Segment name	Field start byte	Field length	Field name (optional)					
0305	Group ID	Group stop TSM (DB2)			Group stop TSM (TOD)	Group stop TSM ID	Group stop TSM creator	Selected flag, TSM creator	PRDS sequence number	PRDS data set name	Update TSM
0400					Select complete flag	Update TSM					

Figure 223. Format of the Selector Control File Records

The Selector control file contains the following records:

0100 (version/release/modification) record

The SCF is created when the IMS DPROP environment is generated, and is primed with the 0100 record. The 0100 record contains the version, release, and modification level of the product. For the current release.

0101 (default SSID) record

The SCF Apply utility ADDSSID control statement creates 0101 records.

Each 0101 record contains an IMS subsystem ID. The set of 0101 records in the SCF, contain the set of default IMS subsystem IDs for the IMS DPROP environment. To determine what online IMS logs need to be processed for a particular propagation group, the Selector needs to know what IMS subsystems, if any, update the databases in the propagation group.

Typically there are few IMS subsystems in use. Therefore, propagation groups are likely to use the same subsystems. You can avoid specifying IMS SSIDs for every propagation group by defining a default set of IMS SSIDs with the SCF Apply utility ADDSSID control statement. Then you must specify SSIDs only for propagation groups that do not use the default.

0200 (database) record

The SCF Apply utility ADDDBASE control statement creates a 0200 record automatically when it adds a database to a propagation group.

Each 0200 record contains the name of an IMS database.

0202 (database/quiesce timestamp) record

The Status Change utility (SCU) CREATETSM control statement creates 0202 records.

Each 0202 record contains a database quiesce timestamp in both ISO DB2 format and TOD format indicating the time the database was quiesced (or offline, in the case of DEDBs).

0300 (propagation group) record

The SCF Apply utility ADDGROUP control statement creates 0300 records.

Each 0300 record contains the name of a IMS DPROP propagation group.

0301 (propagation group SSID) record

Each 0301 record contains the name of an IMS subsystem ID.

To determine what online IMS logs must be processed for a particular propagation group, the Selector needs to know what IMS SSIDs, if any, update the databases in the propagation group.

Typically there are few IMS subsystems in use. Therefore, propagation groups are likely to use the same subsystems. You can avoid specifying IMS SSIDs for every propagation group by defining a default set of IMS SSIDs with the SCF Apply utility ADDSSID control statement.

However, you must use the SCF Apply utility ADDSSID control statement to create a 0301 record when:

- The IMS subsystems that update the databases in a propagation group differ from the default set of IMS subsystem IDs.
- The databases in a propagation group are updated in batch only, In this case use SSID=NONE on the ADDSSID control statement.

If an IMS SSID=NONE is specified for a propagation group, the Selector does not attempt to locate online IMS logs for the group. If IMS SSIDs other than NONE have been specified for a propagation group in 0301 records, the Selector uses the set of IMS SSIDs defined for the group to locate the online IMS logs for the group. Otherwise, the Selector uses the default set of IMS SSIDs defined in the 0101 records to locate the online IMS logs for the group.

0302 (propagation group/database) record

The SCF Apply utility ADDDBASE control statement creates a 0302 record when it adds a database to a propagation group.

A 0302 record is required for each IMS database that is propagated for a particular propagation group.

The 0302 record can also contain a propagation start timestamp for the database. The timestamp assigned as a propagation start timestamp is normally a database quiesce time as defined by a 0202 record. The SCU ASSEGNTSM control statement assigns the propagation start timestamp.

0303 (propagation group/databse/segment) record

The SCF Apply utility ADDSEGM control statement creates the 0303 record.

Each 0303 record contains the name of a segment in an IMS database, to be propagated for a particular propagation group.

If 0303 records exist for a database, the Selector propagates only the segments specified in the 0303 records for the database (segment sensitivity) for that propagation group. Otherwise, it propagates all segments in the database (database sensitivity) for that propagation group.

0304 (propagation group/database/segment/field) record

The SCF Apply utility ADDFIELD control statement creates 0304 records.

Each 0304 record contains the start position of a field within a segment of an IMS database to be propagated for a particular propagation group.

If 0304 records exist for a segment/database, the Selector propagates only the fields specified in the 0304 records for the segment (field Sensitivity) for the propagation group. Otherwise, it propagates all fields in the segment (segment sensitivity) for the propagation group.

0305 (propagation group/stop time) record

You can create 0305 records by:

- Using the SCU CREATETSM control statement
- Using an application program that uses the TSMF callable interface
- Using the Selector

Each 0305 record contains a group stop timestamp (in both ISO DB2 format and TOD format) indicating a time when the Selector is to stop or has stopped propagating for the particular propagation group.

The Selector determines the propagation group stop time based on how the STOP parameter is specified on the SELECT control statement:

A timestamp value is specified on the STOP parameter

The Selector uses the timestamp as the propagation group stop time.

A timestamp ID is specified on the STOP parameter

The Selector searches for a 0305 record for the group with the same TSM ID. If the Selector finds one the Selector did not already use, it uses the 0305 record stop timestamp as the propagation group stop time. Otherwise, the Selector issues an error message and terminates.

INTERIM is specified on the STOP parameter

The Selector determines the stop time for the propagation group and writes a 0305 record to the SCF.

When no STOP parameter is specified on the Selector SELECT control statement

The Selector searches the 0305 records for a stop time that has not been used for the propagation group. If the Selector finds one, it uses the stop timestamp from the 0305 record as the propagation group stop time. Otherwise, the Selector determines the stop time for the propagation group and writes a 0305 record to the SCF.

0400 (selector executing) record

The Selector creates and maintains this record for use in recovery and restart situations.

Chapter 10. The Selector

The Selector component collects propagation log records from IMS log files and writes them to propagation request data sets for later processing by the Receiver component. Input from the Selector control file tell the Selector what IMS log records to retrieve. You can use the SELECT control statement, described on page 323, to name the groups to propagate and to define a timestamp to delimit data propagation.

The following topics provide additional information:

- “Environment”
- “Selector Input and Output ”
- “Selector JCL ” on page 316
- “SELECT control statement ” on page 323
- “Output Messages ” on page 326
- “Return Codes and Error Conditions ” on page 326

Environment

The Selector must exist on the same MVS image as IMS. It runs as an MVS batch job in the following environments:

- Selector and Receiver on the same MVS image or on a different MVS images with shared DASD.

  If the Selector and Receiver are on the same MVS image or different images that share DASD, propagation request data sets (PRDS) produced by the Selector are available to the receiver as soon as they are registered by the PRDS Registration utility (PRU) in the PRDS register table (PRDSREG). You can run the PRDS Registration utility as a job step in the Selector JCL. The Selector automatically generates REGISTER control statements for each PRDS when it writes the trailer record to the PRDS. The Selector writes the REGISTER control statements to the //EKYPRREG file.
- Selector and Receiver on a remote MVS images

  On remote images the propagation requests that are produced by the Selector are not immediately available to the Receiver. You must supply a transport program to send the PRDSs to the receiver.

  All PRDSs must be received by the Receiver site and registered by the PRDS Registration utility before you run the receiver.

If the Selector requires that tapes be mounted, you must request the tape mount through the user attribute data set (UADS) entry for the TSO userid on the job card used to run the PRU, or through RACF. Refer to *TSO Extensions Version 2: Administration* for more information.

Selector Input and Output

Figure 224 on page 314 illustrates the sources of input to the Selector and the output created by the Selector.

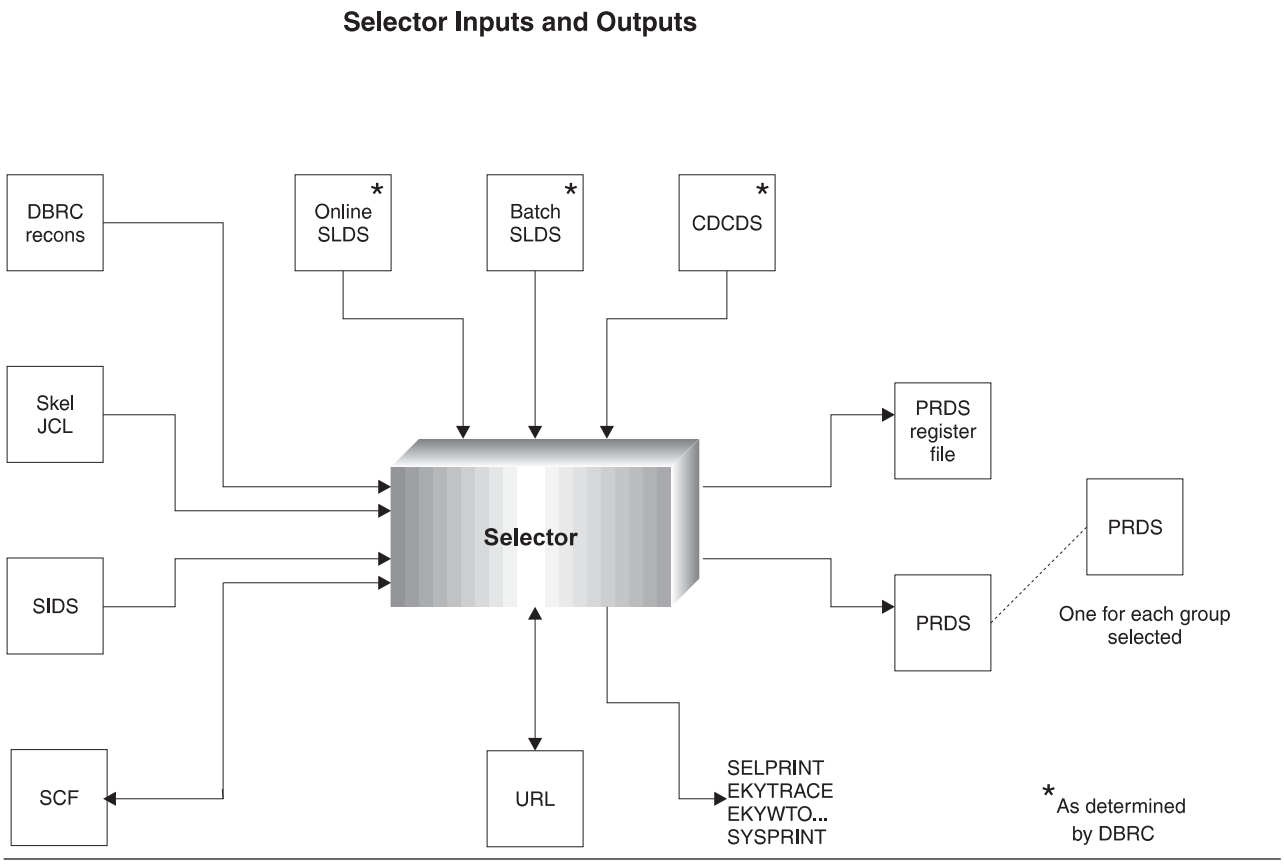

Figure 224. Selector Input and Output

The following list describes the input to the Selector:

Input	Description
SCF	Contains control information that is essential to the operation of the Selector, such as details of propagation groups, databases, segments, fields, and timestamps. For more information on the Selector control file see “SCF Record Layout” on page 307.
SIDS	Contains the Selector control statements that are input by you to specify the processing options used by the Selector. For more information on Selector control statements, see “SELECT control statement ” on page 323.
ULR	Contains uncommitted log records. For more information on the uncommitted log record data set see the appropriate Administrators Guide for your propagation mode.
DBRC RECONS	Contain IMS log information and Changed Data Capture data set information that was located by the DBRC interface. The Selector uses the DBRC interface to retrieve the information from the RECON data sets.
Skel JCL	Contains the skeletal JCL members required to

collect output from database recovery control. This data set is allocated as part of the Selector JCL. The Selector uses the following predefined skeletal members:

EKYSLDS
Contains all system log data set (SLDS) for a particular SSID.

EKYRLDS
Contains all recovery log data sets (RLDS) for a given DBRC qualifier.

EKYCDC1
Contains all PRILOG information created by the modified IMS archive job.

EKYCDC2
Contains all SECLOG information created by the modified IMS archive job.

The following list describes the output from the Selector:

Output	Description
PRDS	The primary output from the Selector. The Selector creates one PRDS for each specified propagation group in the SCF. You must specify the JCL DD statement for the PRDS in the JCL. If you specify SELECT ALL, you must specify a DD name for each propagation group in the SCF. The DD name for each PRDS must correspond to the propagation group. Typically, the PRDS is an MVS generation data group (GDG) data set.
PRDS register file	Contains a propagation request register control statement for each propagation group successfully processed by the Selector.
Selector control file updated	Contains updated Selector control information to indicate that the propagation group was successfully processed and that a new database start time was used.
ULR data set	Is cleaned up by the SCF Apply utility after you delete propagation groups from the SCF. The SCF Apply utility removes any records related to the deleted propagation groups that remain in the ULR data sets. The records are either committed or aborted during the following Selector execution.
Trace files	Contains information used for debugging.
Message files	Contains errors and significant events that are written to the SELPRINT data set.

Selector JCL

This topic describes the following sample JCL:

- EKYUSELA, the sample JCL to allocate the data sets required by the Selector.
- EKYUSELP, the sample JCL to execute the Selector.
- EKYUSELJ, the sample JCL to call the EKYUSELP procedure.

EKYUSELA—Sample JCL to Allocate the Selector Data Sets

Figure 225 shows sample JCL that contains a single step (ALLOC) that creates two data sets that are used by the Selector.

Before you submit the JCL:

- Replace all occurrences of JJJJJJJ with valid job card information.
- Replace all occurrences of `1111111` with the high-level qualifiers IMS DPROP data sets.

```
//JJJJJJJ JOB *** VALID JOB CARD DETAILS ***
//*-------------------------------------------------------------------*
//*     JCL EKYUSELA TO ALLOCATE THE DBRC PRINT FILE AND THE PRDS     *
//*     REGISTER CONTROL FILE (SEE PROCEDURE EKYUSELP)                *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*                                                                  *
//*       CHANGE AS REQUIRED :
//*       ---------------------------------------------------
//*       JJJJJJJ   = JOB NAME
//*       1111111   = HIGH LEVEL QUALIFIER FOR DATASETS CREATED BY
//*                   THIS JOB
//*-------------------------------------------------------------------*
//*       DEPENDENCIES :
//*       ----------------------------------------
//*       NONE
//*
//ALLOC       EXEC PGM=IEFBR14
//*-------------------------------------------------------------------*
//* JCLOUT IS THE DBRC SYSPRINT DATASET FOR THE SELECTOR
//*-------------------------------------------------------------------
//JCLOUT      DD   DSN=1111111.JCLOUT,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(800,(10,1)),
//            DCB=(LRECL=80,BLKSIZE=800,RECFM=FB)
//*-------------------------------------------------------------------*
//* EKYPRREG IS THE PRDS REGISTER FILE OF CONTROL STATEMENTS
//*-------------------------------------------------------------------*
//EKYPRREG    DD   DSN=1111111.EKYPRREG,DISP=(NEW,CATLG),
//            UNIT=SYSDA,SPACE=(800,(10,1)),
//            DCB=(LRECL=80,BLKSIZE=800,RECFM=FB)
```

Figure 225. EKYUSELA: Sample JCL to Allocate the Selector Data Sets

The sample in Figure 225 on page 316 contains the following DD statements:

//JCLOUT
Used by the Selector to store the output from DBRC (JCLOUT).

//EKYPRREG
Contains a PRDS register control statement for each propagation group the Selector processes successfully. If the Selector and Receiver are on the same MVS image, or share DASD between MVS images, the PRDS registration utility (PRU) can use this data set directly to register PRDSs. If the Selector and Receiver are on different MVS images you can use this data set to initiate sending PRDSs from the Selector site to the Receiver site.

EKYUSELP—Sample JCL to Execute the Selector

The sample JCL shown in Figure 226 on page 318 contains a single step that executes the Selector.

Before invoking the procedure, you must replace `1111111` with the high-level qualifiers for the IMS DPROP RESLIB data set.

If you want to specify alternative DBRC RECON data sets, remove comment tags from the &RECON1, &RECON2, and &RECON3 DD statements, and specify the data set names of the required DBRC RECONs.

```
//*-------------------------------------------------------------------*
//*  PROCEDURE EKYUSELP TO EXECUTE THE SELECTOR                 *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                               *
//**        LICENSED MATERIALS - PROPERTY OF IBM                   *
//**                                                               *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                 *
//**                                                               *
//**                                                               *
//**        SEE COPYRIGHT INSTRUCTIONS                             *
//**                                                               *
//*******************************************************************
//*******************************************************************
//*
//*       CHANGE AS REQUIRED :
//*       ---------------------------------------------------
//*
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S STANDARDS :
//*-----------------------------------------------------------------
//*       1111111   = HIGH LEVEL QUALIFIER FOR DPROP RESLIB DATASET
//*
//*-------------------------------------------------------------------*
//*       DEPENDENCIES :
//*       ----------------------------------------
//*       EKYUPRDA     - CREATE GDG FOR EACH PRDS
//*       EKYUSELA     - ALLOCATE SELECTOR DATASETS
//*-------------------------------------------------------------------*
//*
//EKYUSELP PROC EKYPREF=1111111
//*-----------------------------------------------------------------
//*   RUN THE SELECTOR
//*-----------------------------------------------------------------
//SELECTOR   EXEC PGM=EKYB000X
//EKYRESLB   DD   DSN=&EKYPREF..EKYRESLB,DISP=SHR
//EKYSCF     DD   DSN=&EKYPREF..EKYSCF,DISP=SHR
//EKYULR     DD   DSN=&EKYPREF..EKYULR,DISP=SHR
//EKYPRREG   DD   DSN=&EKYPREF..EKYPRREG,DISP=MOD
//SELPRINT   DD   SYSOUT=*
```

Figure 226. EKYUSELP: Sample JCL to Execute the Selector (Part 1 of 2)

```
//*-----------------------------------------------------------------
//*---   DATASETS REQUIRED BY DBRC INTERFACE
//*-----------------------------------------------------------------
//SYSIN      DD   DSN=&&SYSIN,UNIT=SYSDA,
//           SPACE=(800,(10,1),RLSE),
//           DCB=(LRECL=80,BLKSIZE=800,RECFM=FB)
//SYSPRINT   DD   DSN=&&SYSPRINT,UNIT=SYSDA,
//           SPACE=(1330,(10,1),RLSE),
//           DCB=(LRECL=133,BLKSIZE=1330,RECFM=FBA)
//EKYSDMEM   DD   DSN=&EKYPREF..EKYSKEL,DISP=SHR
//JCLOUT   DD   DSN=&EKYPREF..JCLOUT,DISP=SHR
//*-----------------------------------------------------------------
//*       ADD THE RECON DD STATEMENTS HERE IF YOU ARE NOT USING
//*       DYNAMIC ALLOCATION
//*----------------------------------------------------------------
//*       &RECON1   = --+
//*       &RECON2   =   '  -  DBRC RECON DATASETS
//*       &RECON3   = --+
//*
//*
```

Figure 226. EKYUSELP: Sample JCL to Execute the Selector (Part 2 of 2)

The //STEPLIB, //EKYRESLB, and //EKYPRINT DD statements are common to many IMS DPROP components and are described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The remaining statements in Figure 226 on page 318 are described in the following list:

PROC statement
Defines the various keyword parameters in the JCL, including the high level qualifiers for the IMS DPROP library data set names.

EXEC statement
Calls the Selector program, EKYB000X.

//EKYSCF DD statement
Contains control information that is required by the Selector, such as details of propagation groups, databases, segments, fields and timestamps. The SCF is a VSAM keyed sequential data set that is created and initialized during IMS DPROP installation and written to by the SCF apply job, the Selector, and the timestamp marker utility under the SCU. The SCF is updated by the Selector to indicate that propagation groups are processed successfully and that database new start times are used.

//EKYULR DD statement
Contains IMS log records that were not committed during the Selector execution. These log records are committed or aborted during the next execution of the Selector. The ULR is a VSAM keyed sequential data set (KSDS) that is created and initialized during IMS DPROP installation.

//EKYPRREG DD statement
Contains a PRDS register control statement for each propagation group the Selector processed successfully.

The Selector procedure, EKYUSELP, replaces the contents of the //EKYPRREG data set each time the Selector runs if you specify DISP=OLD. If you want to keep previous PRDS register entries, change the //EKYPRREG DD statement to DISP=MOD.

//SELPRINT DD statement
Contains activity and errors recorded during the Selector execution.

//SYSIN DD statement
Contains generated input commands to DBRC. This DD statement must be allocated to a writable data set and *not* allocated as instream JCL.

//SYSPRINT DD statement
Contains the print output from DBRC. This DD statement must be allocated to a readable data set and *not* allocated as a SYSOUT file.

//EKYSDMEM DD statement
Contains the DBRC skeleton JCL library required to support the DBRC GENJCL.USER commands.

//JCLOUT DD statement
Contains the output from DBRC (JCLOUT).

//RECON1...//RECON3
Used to specify alternative RECONS. If you do not specify the RECONS DD statements, the default set for the system is used. To specify alternative RECONS, remove the comment tags from the DD statements in the procedure and specify the data set names of the DBRC RECONS you want to use.

Because the SCF and the ULR data sets are both read from and written to each time the Selector executes, you must specify the same data set names for successive executions of a Selector.

EKYUSELJ—Sample JCL to Call the Selector

The sample JCL shown in Figure 227 on page 321 contains the following steps:

- RUNSEL invokes the procedure, EKYUSELP, which executes the Selector.
- CATALOG catalogs the PRDSs if the Selector completes successfully (return code of 0 or 4).

Before you submit the JCL:

- Replace all occurrences of `JJJJJJJ` with valid job card information.
- Replace all occurrences of `1111111` with the high-level qualifiers for the IMS DPROP data sets.
- Replace all occurrences of `2222222` with the number of blocks to allocate for all PRDSs. Allocate enough space to accommodate the maximum number of IMS propagation records you expect to be selected for any propagation group. See the appropriate Administrators Guide for your propagation mode for more information on storage requirements.
- Replace all occurrences of `3333333` with the name of the propagation group for which you want to select propagation log records. If you want to specify more than one propagation group, you must specify each one with a separate DD statement.

```
//JJJJJJJ  JOB  *** VALID JOB CARD DETAILS ***
//*-------------------------------------------------------------------*
//*  JCL EKYUSELJ TO RUN DATA PROPAGATOR - SELECTOR (EKYUSELP)        *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*  MODIFY THE FOLLOWING VALUES TO SUIT YOUR INSTALLATION           *
//*                                                                  *
//*  JJJJJJJ  - A VALID JOB NAME                                     *
//*                                                                  *
//*  1111111  - THE PREFIX USED FOR DPROP DATA SETS                  *
//*                                                                  *
//*  2222222  - THE NO. OF BLOCKS TO ALLOCATE FOR A PRDS. THE        *
//*             AMOUNT OF SPACE ALLOCATED SHOULD CATER FOR THE       *
//*             NUMBER OF PROPAGATION RECORDS SELECTED FOR A GROUP   *
//*                                                                  *
//*  3333333  - THE GROUP NAME FOR WHICH PROPAGATION RECORDS ARE     *
//*             TO BE SELECTED. IF MULTIPLE NAMES ARE SPECIFIED,     *
//*             ONE DD STATEMENT PER GROUP NAME IS REQUIRED.         *
//*                                                                  *
//*             NOTE: IF 'SELECT=ALL' IS SPECIFIED, A DD STATEMENT   *
//*                   FOR EVERY REGISTERED GROUP IS REQUIRED.        *
//*                                                                  *
//*  ALTER THE CONTROL STATEMENTS FOR THE EKYSIDS AND EKYIN FILES    *
//*  TO SUIT YOUR REQUIREMENTS                                       *
//*                                                                  *
//*  FOR EACH GROUP TO BE SELECTED :-                                *
//*                                                                  *
//*       - SPECIFY THE GROUP ID IN A SELECT CONTROL STATEMENT       *
//*         IN THE //EKYSIDS DD STATEMENT                            *
//*                                                                  *
//*       - IN THE RUNSEL AND CATALOG STEPS, ADD A DD STATEMENT FOR  *
//*         THE GROUP. THE PRDS DDNAME MUST MATCH THE GROUP ID IN THE *
//*         SCF. IT IS ALSO RECOMMENDED THAT THE GROUP NAME BE       *
//*         INCLUDED IN THE GDG BASE NAME.                           *
//*                                                                  *
//*-------------------------------------------------------------------*
//*
```

Figure 227. EKYUSELJ: Sample JCL to Call the Selector Procedure (Part 1 of 2)

```
//RUNSEL  EXEC PROC=EKYUSELP
//*
//EKYSIDS  DD   *
 SELECT GROUP=3333333;
/*
/*
//3333333  DD DSN=1111111.PRDS.3333333(+1),DISP=(NEW,PASS),
//         SPACE=(32760,(2222222),RLSE)
//*
//*-------------------------------------------------------------------
//* CATALOG THE PRDS FOR EACH GROUP
//*
//* NOTE: THE CATALOG STEP MUST BE RUN IN ORDER TO CATALOG THE PRDS
//*     DATASET(S) AND MAKE THEM AVAILABLE TO THE PRU AND THE RECEIVER.
//*     THIS STEP SHOULD ONLY EXECUTE AFTER RC 0 OR RC 4 SELECTOR
//*     RUNS (VALID PRDSS ARE CREATED IN THESE CASES). DO NOT CATALOG
//*     PRDSS RESULTING FROM RC 8 OR HIGHER SELECTOR RUNS AS THESE ARE
//*     INVALID DATASETS AND CANNOT BE PROCESSED BY THE PRU OR THE
//*     RECEIVER.
//*
//* YOU MUST PROVIDE A DD STATEMENT IN THIS STEP FOR EACH PRDS THAT
//* IS SPECIFIED IN THE SELECTOR STEP. IF 'SELECT=ALL' WAS SPECIFIED
//* FOR THE SELECTOR, A DD STATEMENT FOR EVERY REGISTERED GROUP MUST
//* INCLUDED IN THIS STEP.
//*
//*-------------------------------------------------------------------
//CATALOG     EXEC PGM=IEFBR14,COND=(8,LE,RUNSEL.SELECTOR)
//3333333     DD   DSN=1111111.PRDS.3333333(+1),DISP=(OLD,CATLG)
//*-------------------------------------------------------------------
//*
```

Figure 227. EKYUSELJ: Sample JCL to Call the Selector Procedure (Part 2 of 2)

The sample in Figure 227 contains the following statements:

//EKYSIDS DD statement

Contains the Selector input control statements that specify the groups to be selected. For further information on the SELECT control statement, refer to “SELECT control statement ” on page 323.

***3333333*(PRDS)**

Contains the IMS propagation log records selected for a propagation group. Specify one DD statement specified for *each* propagation group. The DDNAME must match the group identifier in the SCF.

The PRDS can be a member of a previously defined GDG or a sequential file. Use MVS GDGs for the PRDSs, so you can use the Selector JCL repeatedly without change.

In the Selector JCL, the PRDS is referred to by the GDG relative index. A new set of data sets is created for each execution of the Selector. The Selector always creates the (+1) version for the GDG, and uses the REGISTER control statement to write the data set name to EKYPREG.

The GDG is catalogued in the second step of the job, only if the Selector completes successfully (return code of 0 or 4).

SELECT control statement

You input SELECT control statements directly to the EKYSIDS data set of the Selector JCL. See Figure 227 on page 321 for an example of the Selector JCL.

Use the SELECT control statement to define the execution options for the Selector.

Syntax of the SELECT control statement

Figure 228 shows the syntax of the SELECT control statement.

```
                          ┌─,──────────┐
                          ▼            │
►►─SELECT─┬─GROUP=(──────group_id──────┴─)─┬──────────────────────────────────────────────►◄
          └─ALL──────────────────────────┘ │                                              │
                                           └─,─STOP─=─┬─┌─INTERIM─┐─┬─┬──────────────────┬─┬─┘
                                                      │ │          │ │ │                  │ │
                                                      │ └──────────┘ │ │                  │ │
                                                      ├─timestamp────┤ │                  │ │
                                                      └─TSM,ID=tsm_id┘ │                  │ │
                                                                       │                  │ │
►►─┬──────────────────────────────────────────────────────────────────┴──────────────────┴─┬─;──────────────────────────────────────────────►◄
   │                                                                                        │
   └─,─INVUOW─=─┬─┌─STOP─┐─┬─┬──────────┬─┬─┘
                │ │       │ │ │          │ │
                │ └───────┘ │ │          │ │
                └─IGNORE────┘ │          │ │
                              │          │ │
►►─┬──────────────────────────┴──────────┴─┬──────────────────────────────────────────────►◄
   │                                       │
   └─,─QMGR─=─queue_manager name─,─QUEUE─=─queue_name─┘
```

Figure 228. The SELECT control statement

GROUP=(*group_id,group_id,...* **)**
Specifies the propagation groups to be processed on the current Selector execution. If you specify a value on the STOP parameter, the value applies to all *group_id*s specified on the control statement. If you require different stop times for some propagation groups, you must specify additional SELECT control statements.

ALL
Selects all propagation groups in the SCF. If you specify a value on the STOP parameter, the value applies to all propagation groups in the SCF.

If you specify this parameter:

- Do not specify additional SELECT control statements; if you do, the Selector terminates when you try to execute it.
- Add a DD statement to the Selector JCL for each propagation group defined in the SCF.

STOP= *timestamp* **|** **TSM,ID=***tsm_id* **|** **INTERIM**
Specifies a stop time delimiter for selecting source data to propagate. You can use a stop time value in any of the following formats, or you can specify no stop time value.

timestamp
Specifies that the Selector use a user-supplied timestamp in ISO/DB2 format as a stop time value. This timestamp overrides any other timestamps

that were created for the propagation groups on the propagation group list. An example of a user-defined timestamp is:

```
1994-06-01-11.48.23.169268
```

The format of this example is *YYYY-MM-DD-HH.MM.SS.NNNNNN,* where:

YYYY	Represents the four digits of the year
MM	Represents the month (include zeros)
DD	Represents the day (include zeros)
HH	Represents the hour (include zeros)
MM	Represents the minutes (include zeros)
SS	Represents the seconds (include zeros)
NNNNNN	Represents the microseconds

TSM
Specifies that the Selector use the timestamp associated with the TSM ID specified on the ID parameter as the stop time for the specified propagation groups.

ID= *tsm_id*
Specifies the required TSM ID.

The Selector searches the SCF for the latest 0305 (propagation group/stop time) record that has a TSM ID that corresponds to the specified *tsm_id*. and that has not already been selected. If no such 0305 (propagation group/stop time) record exists, the Selector issues an error message and terminates.

You can use this parameter to select a particular stop timestamp from several that were created for the specified propagation groups.

INTERIM
Specifies that the Selector determine the stop time value based on the available log data. If other 0305 records exist for these propagation groups, they are ignored and the Selector uses the 0305 record it created for the stop time value it determined.

Note: If you do not specify a stop time value on the SELECT control statement, the most recently created 0305 record is used. If neither a stop time value on the SELECT control statement nor a 0305 record exists, then the default value is INTERIM, and the Selector determines the stop time value and creates its own 0305 record.

INVUOW= STOP | IGNORE
Specifies how the Selector is to respond if it finds that a source UOW is missing a *first* record.

STOP
Specifies that the Selector terminate if it finds that the first 9904 (update) record for a UOW is not flagged as the first record. This value is the default for INVUOW.

IGNORE
Specifies that the Selector continue processing even if it finds an invalid UOW. If the Selector finds an invalid UOW, it:

- Writes the recovery token for the invalid UOW to the trace data set
- Writes a message to the SELPRINT data set

- Flags the condition in the Selector activity listing

Examples of the SELECT control statement

Example 1 – SELECT With a User-Defined Timestamp

The example in Figure 229 selects the GROUP01 and GROUP02 propagation groups with a user-defined timestamp as a stop time value. The IGNORE parameter directs the Selector to ignore any invalid units of work that it finds. A user-defined timestamp can cause an invalid unit of work to occur.

```
SELECT GROUP=(GROUP01,GROUP02),STOP=1994-08-06-10.30.20.143568,INVUOW=IGNORE
```

Figure 229. SELECT control statement Example 1

Example 2 – SELECT with a TSM ID

The example in Figure 230 selects the GROUP02 and GROUP03 propagation groups using TSM, as a stop time value. The IGNORE parameter directs the Selector to ignore any invalid unit of works that it finds. If the SCF contains more than one 0305 (propagation group/stop time) record with the specified TSM ID, the Selector uses the latest one that has not already been used.

```
SELECT GROUP=(GROUP02,GROUP03),STOP=TSM,ID=tsmid2,INVUOW=IGNORE
```

Figure 230. SELECT control statement Example 2

Example 3 – SELECT with INTERIM

The example in Figure 231 selects the GROUP03 and GROUP04 propagation groups using the INTERIM parameter as a stop time value. The Selector ignores any timestamps that exist for the propagation groups in 0305 records in the SCF, and creates its own 0305 record based on the stop time value it determined. This example uses the default value, INVUOW=STOP. An invalid unit of work is less likely to occur when the Selector determines the stop time value.

```
SELECT GROUP=(GROUP03,GROUP04),STOP=INTERIM,INVUOW=STOP
```

Figure 231. SELECT control statement Example 3

Example 4 – SELECT ALL

The example in Figure 232 selects all propagation groups in the SCF and uses a Selector determined stop time.

```
SELECT ALL,STOP=INTERIM;
```

Figure 232. SELECT control statement Example 4

Example 5 – SELECT with No Timestamp

The example in Figure 233 on page 326 selects GROUP01. The Selector searches the SCF for a 0305 (propagation group/stop time) record. It selects GROUP01 and uses the associated timestamp as the stop timestamp for the propagation group. If

the Selector finds more than one 0305 record for the propagation group, it uses the latest one that was not previously used. If the Selector does not find a 0305 record for the propagation group, it determines the stop time for the propagation group.

```
SELECT GROUP=(GROUP01),INVUOW=IGNORE
```

Figure 233. SELECT control statement Example 5

Output Messages

Error messages for the Selector begin with EKYB. For more information on Selector messages and codes, see *IMS DPROP Messages and Codes*.

Return Codes and Error Conditions

The Selector provides the return and reason codes shown in Table 34.

Table 34. Selector Return and Reason Codes

Return Code	Reason Code	Description
0	0	The Selector completed processing and terminated normally. Information messages are issued.
4	0	A minor processing error occurred. A warning message is issued, and processing continues. Some examples are: • A database new start record (SCF 0302) contains a timestamp later than the stop time for the group. • An IMS ROLS record was read for which no corresponding SETS record could be found.
4	4	A minor processing error occurred. A warning message is issued and processing continues. An example is: • There are no IMS log files to process that satisfy the selection criteria.
8	8	A major processing error occurred. An error message is issued, and processing terminates. Examples include: • Invalid data was specified on SELECT control statement. • Errors occurred during date or time conversion. • The VSAM data set is empty when at least one record is expected (either SCF or ULR data sets). • The Selector stop time is zero.
8	32	A major processing error occurred while attempting to access a data set used by the Selector. An error message is issued and processing terminates.
16		A severe processing error occurred. An error message is issued, and processing terminates.
16	0	A data set allocation error occurred, dynamically allocating, or opening or closing one of the following data sets: • PRDS • Selector control file • ULR data set • IMS log file (SLDS, CDCDS)

Table 34. Selector Return and Reason Codes (continued)

Return Code	Reason Code	Description
16	4	A data set access error occurred while reading from or writing to one of the following data sets: • PRDS • Selector control file • ULR data set • IMS log file (SLDS, CDCDS)
16	24	The Selector detected that an IMS unit of work was missing a first record (inquiry data), and the INVUOW keyword on the SELECT control statement was set to STOP.
16	32	A data integrity error occurred. An example is: • An IMS 9904 log record that is not in the expected format.
16	36	Invalid data has been passed to one Selector module from another one. An example is: • An internal control block does not contain the expected information.
16	40	An internal Selector control block was corrupted.

Chapter 11. The Receiver

Execute the Receiver together with the Selector to update the DB2 tables. The Selector collects the IMS propagation log records and then passes them to the Receiver as propagation request data sets (PRDS). The Receiver retrieves the PRDSs and passes them to the RUP, which uses them to update the DB2 tables.

This topic describes the:
- Environment in which the Receiver runs
- JCL to call and execute the Receiver
- Receiver control statement
- PRDS control statement

The following topics provide additional information:
- “Environment”
- “Receiver JCL ”
- “RECEIVER control statement ” on page 332
- “PRDS control statement ” on page 333

Environment

The Receiver runs as an MVS batch job and can be executed in the following environments:

- Receiver and Selector on a local MVS image or on a local MVS image with shared DASD.

  On a local MVS image the PRDSs that the Selector produces are available to the Receiver as soon as they are registered by the PRU in the PRDS registration table. You can run the PRU as a job step in the Selector JCL.

  The Receiver decides which PRDSs to read from the PRDS registration table based on the Receiver groups specified in the RIDS and on the propagation request control table and the Receiver control table. The Receiver applies the appropriate PRDSs to the DB2 tables.
- Receiver and Selector on a remote MVS image.

  On a remote MVS image the PRDSs that are produced by the Selector are not immediately available to the Receiver. In this case, you must supply a transport program to send the PRDSs to the receiver.

  All PRDSs must be received by the Receiver site and registered by the PRU before the Receiver can run.

If the Receiver requires tapes, you must request that the tape volumes be mounted. You request tape mounts through the user attribute data set (UADS) entry for the TSO userid on the job card used to run the PRU, or through RACF. For more information, see to *TSO Extensions Version 2: Administration*.

Receiver JCL

This topic describes the following JCL samples:
- EKYURCVP, the JCL to execute the Receiver
- EKYURCVJ, the JCL to call EKYUPCVP

EKYURCVP - Sample JCL to Execute the Receiver

Figure 234 shows the sample JCL for EKYURCVP, to execute the Receiver. Before you invoke the procedure:

- Replace JJJJJJJ with job name information.
- Replace 1111111 with the high-level qualifiers for the IMS DPROP RESLIB.

```
JJJJJJJ     *****VALID JOB CARD DETAILS******
//*-------------------------------------------------------------------*
//*       PROCEDURE EKYURCVP TO EXECUTE THE RECEIVER JOB              *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S NEED:          *
//*  -------------------------------------------------------          *
//*  JJJJJJJ  = JOB NAME                                              *
//*  1111111  = HIGH LEVEL QUALIFIER(S) OF DPROP RESLIB               *
//*                                                                   *
//*-------------------------------------------------------------------*
//*
//*
//EKYURCVP   PROC EKYPREF=1111111
//*
//RUN        EXEC PGM=EKYF000X
//EKYRESLB   DD   DSN=&EKYPREF..EKYRESLB,DISP=SHR
//*DSNTRACE   DD   SYSOUT=*
//*
```

Figure 234. EKYURCVP: Sample JCL to Execute the Receiver

The DD statements //STEPLIB, //EKYRESLB, //EKYPRINT, and //EKYAUDIT are common to many IMS DPROP components and are described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The STEPLIB must include the name of the SQL update module library that contains the propagation requests to be processed by the Receiver. The remaining statements in Figure 234 are described in the following list:

PROC statement
Defines the various keyword parameters of the JCL procedure.

EXEC statement
Calls the Receiver program (EKYF000X).

//DSNTRACE DD statement
Optional. Contains trace data produced by the DB2 call attach facility (CAF) that is used by the Receiver.

EKYURCVJ - Sample JCL to Call EKYURCVP

Before you invoke the procedure you must:

- Replace JJJJJJJ with job name information.

- Replace 1111111 with the Receiver name.
- Replace 2222222 with the DB2 subsystem identifier.
- Replace 3333333 with the DB2 plan name for this execution.
- Replace 4444444 with the number of units of work for each DB2 commit.
- Replace 5555555 with the PRDS group name.
- Replace 6666666 with the PRDS sequence number.
- Replace 7777777 with the Receiver stop criteria.
- Replace 8888888 with the TSM identifier of the stoptime.

Figure 235 shows the sample JCL to call EKYURCVP.

```
//JJJJJJJ JOB     *** VALID JOB CARD DETAILS ***
//*
//*-------------------------------------------------------------------*
//*       JOB EKYURCVJ TO EXECUTE THE EKYURCVP PROCEDURE              *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*    ADAPT THE JCL AS FOLLOWS :
//*    --------------------------
//*
//*   JJJJJJJ  - JOB NAME
//*
//*   ALTER THE CONTROL STATEMENTS IN THE EKYRIDS INPUT FILE TO
//*   SUIT YOUR REQUIREMENTS
//*
//*   1111111  -  RECEIVER NAME THAT IS TO PROCESS THIS PRDS
//*   2222222  -  DB2 SUBSYSTEM IDENTIFIER
//*   3333333  -  THE DB2 PLAN NAME TO BE USED FOR THIS EXECUTION
//*   4444444  -  THE NUMBER OF UOWS FOR EACH DB2 COMMIT
//*   5555555  -  THE PRDS GROUP NAME
//*   6666666  -  THE PRDS SEQUENCE NUMBER (OPTIONAL)
//*   7777777  -  THE RECEIVER STOP CRITERIA(TIMESTAMP/TSM/END)
//*   8888888  -  THE TSM IDENTIFIER OF THE STOPTIME (OPTIONAL)
//*
//*******************************************************************
//RUNRCVR  EXEC EKYURCVP
//EKYRIDS  DD   *
  RECEIVER NAME=1111111,DB2SSID=2222222,PLAN=3333333,COMMCNT=4444444;
  PRDS GROUP=5555555,NEXTPRDS=6666666,STOP=7777777,ID=8888888;
/*
//*
```

Figure 235. EKYURCVJ: Sample JCL to Call EKYURCVP

The //EKYRIDS DD statement provides the Receiver control statements. The Receiver processes only the Receiver name specified on the these control statements. The Receiver control statements consist of a RECEIVER control statement and a PRDS control statement. One and only one of these control statements must exist for each execution of the Receiver.

RECEIVER control statement

Use the RECEIVER control statement to specify:

- The name of the Receiver that will process a PRDS
- The name of the DB2 subsystem to be accessed and the DB2 plan name
- The number of committed unit of works to process before a DB2 commit is issued

Syntax of the Receiver control statement

Figure 236 shows the syntax of the RECEIVER control statement.

```
►►──RECEIVER NAME=receiver_name──┬──────────────────────────────────────┬──────────►◄
                                 └──,──DB2SSID──=──db2_subsystem_name──┘

►►──┬──────────────────────────────────────────────────────────────────┬──;──────►◄
    │                                                                  │
    │                                                   ┌─1─┐          │
    └──,──PLAN──=──plan_name──,──COMMCNT──=──┬──────┬──┘
                                             └──n───┘
```

Figure 236. The Receiver control statement

NAME= *receiver_name*
Specifies the Receiver that is to process a propagation request data set. The Receiver name is used as the primary key in the RCT and consists of 1 to 8 characters.

DB2SSID= *db2_subsystem*
Specifies the MVS subsystem name of the DB2 system to be accessed. If this parameter is not supplied, then IMS DPROP uses the default subsystem identifier in module DSNHDECP.

PLAN= *plan_name*
Specifies the name of the DB2 plan to use in this Receiver execution.

COMMCNT= **1** | *n*
Specifies the number of committed source units of work to apply prior to issuing a DB2 commit.

1 Specifies that an SQL commit is to be issued for every committed source unit of work applied to DB2. This value is the default for COMMCNT.

n Specifies that an SQL commit is to be issued after *n* committed source units of work are applied to DB2, (*n* is any integer in the range 1 to 99999999 inclusive).

Examples of RECEIVER control statements

Example 1

The example in Figure 237 on page 333 is a RECEIVER statement with a commit count of 1. It executes the Receiver named REC2 in the default DB2 subsystem, has a plan name of RXPLAN, and causes the Receiver is to issue an SQL commit for every source committed unit of work applied to DB2.

```
RECEIVER NAME=REC1,PLAN=RXPLAN,COMMCNT=1
```

Figure 237. RECEIVER control statement: Example 1

Example 2

The example in Figure 237 is a RECEIVER statement with a commit count of 4. It executes the Receiver named REC2 in the default DB2 subsystem, uses a plan name of RXPLAN, and causes the Receiver to issue an SQL commit after 4 source committed units of work are applied to DB2.

```
RECEIVER NAME=REC2,DB2SSID=DSNX,PLAN=RXPLAN,COMMCNT=4
```

Figure 238. RECEIVER control statement: Example 2

PRDS control statement

The PRDS control statement describes the source PRDS for a particular execution of the Receiver.

Syntax of the PRDS control statement

Figure 239 shows the syntax of the PRDS control statement.

```
►►──PRDS GROUP=group_id──┬──────────────────────┬──────────────────────────────►◄
                         └─,──NEXTPRDS──=──n────┘

►►──┬──────────────────────────────────────────┬──;──────────────────────────►◄
    └─,──STOP──=──┬──END──────────────────────┬┘
                  ├──timestamp────────────────┤
                  └──TSM,ID=tsm_id────────────┘
```

Figure 239. The PRDS control statement

GROUP= *group_id*
Specifies the group of PRDSs to link to the Receiver.

NEXTPRDS= *n*
Specifies that the Receiver process the PRDS with the sequence number specified by *n* (*n* is any integer between 1 and 999999999 inclusive).

The Selector creates multiple PRDSs for a propagation group and assigns a unique sequence number to each one. The Receiver determines the sequence of PRDSs to be received, but you can override it with this optional parameter.

STOP= **END** | *timestamp* | **TSM,ID=***tsm_id*
Informs the Receiver of the time at which it is to stop applying PRDSs to the DB2 tables.

END
Specifies that all registered PRDSs starting from the current sequence number be processed. For example, if the next PRDS sequence number is 5, and PRDSs 5, 6, and 7 are marked as available in the PRDS register table, the Receiver applies all three of these PRDSs. This value is the default for STOP.

timestamp
An example of a user-defined timestamp is:

```
1994-03-03-11.48.23.169268
```

The format is *YYYY-MM-DD-HH.MM.SS.NNNNNN*, where:

YYYY	Represents the four digits of the year
MM	Represents the month (include zeros)
DD	Represents the day (include zeros)
HH	Represents the hour (include zeros)
MM	Represents the minutes (include zeros)
SS	Represents the seconds (include zeros)
NNNNNN	Represents the microseconds

TSM,ID= *tsm_id*
Specifies that the Receiver terminate processing if *tsm_id* is found in the PRDS trailer record.

A TSM ID is an 8 byte character value that is used as a substitute for an actual timestamp value. Using this identifier makes it easier to reference timestamps.

Examples of a PRDS Statements

Example 1

The example in Figure 240 is a PRDS statement using END. It applies all occurrences of the GROUP01 PRDS in the PRDS register table from the current sequence number to DB2 tables:

```
PRDS GROUP=GROUP01,STOP=END
```

Figure 240. PRDS control statement: Example 1

Example 2

The example in Figure 241 is a PRDS statement using NEXTPRDS and a timestamp. It processes the GROUP2 PRDS with a sequence number of 5. The Receiver stops applying updates when it encounters a unit of work whose commit time is greater than 11.20 on August 10 1994:

```
PRDS GROUP=GROUP02,NEXTPRDS=5,STOP=1994-08-10-11.20.1435 68
```

Figure 241. PRDS control statement: Example 2

Example 3

The example in Figure 242 is a PRDS statement using a TSM ID. It applies all occurrences of the GROUP03 PRDS in the PRDS registration table from the current sequence number to DB2 tables until it encounters a trailer record containing the `EOD` TSM ID.

```
PRDS GROUP=GROUP01,STOP=TSM,ID=EOD
```

Figure 242. PRDS control statement: Example 3

Chapter 12. PRDS Registration Utility

The PRDS Registration utility (PRU) maintains the PRDS register table and allows users to view information on the PRDS register table. The PRDS register table is one of the IMS DPROP Directory Tables.

The following topics provide additional information:

Overview

Use the PRU to:

- Register PRDSs in the PRDS register table.
- Unregister PRDSs from the PRDS register table.
- List PRDSs in the PRDS register table.

PRDSs are created by the Selector and contain the propagation log records created by IMS to be applied to the corresponding DB2 tables by one or more Receivers. The Selector creates one PRDS for each selected propagation group during each execution.

The PRDS Registration utility uses the information in the PRDS header record to enter the PRDS in the PRDS register table. The PRDS register table is a DB2 table and contains all the PRDS entries for each Receiver. The PRDS register table is the interface between the PRDS Registration utility and the Receiver. For more information on how a PRDS is created, and the contents of a PRDS, see the appropriate Administrators Guide for your propagation mode.

Execute the PRU prior to running the Receiver so that there are entries in the PRDS register table for the Receiver to process. The Receiver determines from the PRDS register table what PRDSs to read, and subsequently applies the PRDSs to the DB2 target tables. See “The PRDS Register Table (DPRPRDSR) ” on page 163 for more information about the PRDS register table.

Input and Output

Figure 243 on page 336 illustrates the sources of input to the PRDS Registration utility and the output created by this program.

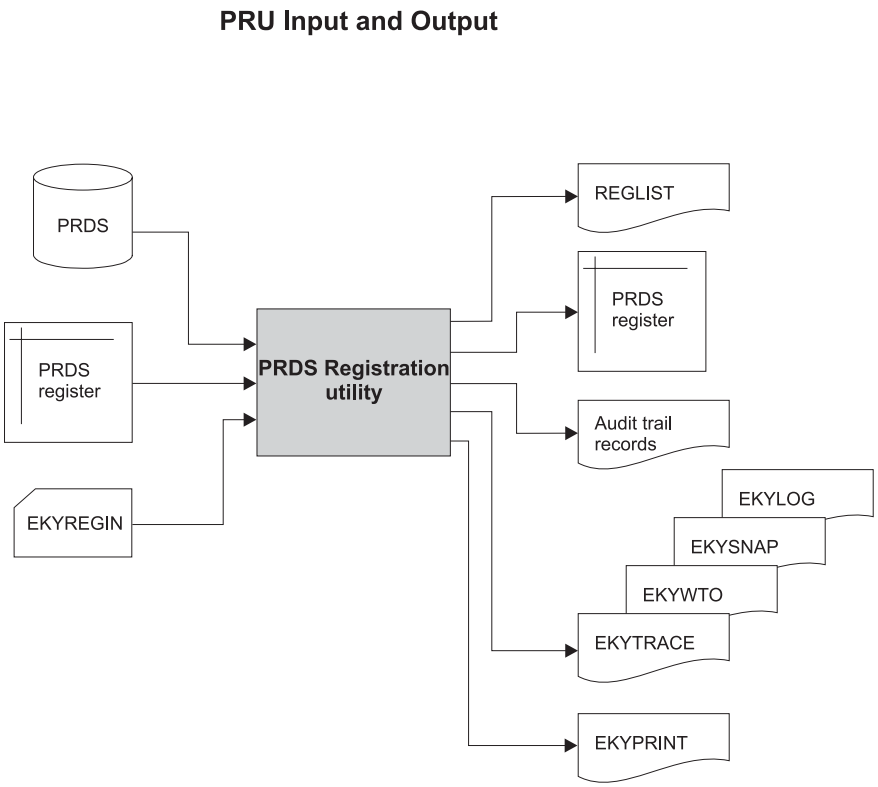

Figure 243. PRU Input and Output

The input to the PRDS Registration utility consists of:

- The PRDS register table.
- The PRDS data set.
- The //EKYREGIN file containing the three types of PRDS Registration utility control statements entered individually, or the //EKYPRREG file that contains REGISTER control statements only.

The output from the PRDS Registration utility consists of:

- The updated PRDS register table.
- The //REGLIST file, which contains the output, if any, from the LIST control statement and the output, if any, from the UNREGISTER control statement.
- The //EKYTRACE, //EKYWTO, //EKYSNAP and //EKYLOG files. These files are optional, refer to the IMS DPROP Diagnosis for details.
- The //EKYPRINT file, which contains any errors or significant events.
- Audit trail records, which indicate the success of PRDS Registration utility operations. The PRDS Registration utility writes audit trail records when a PRDS is successfully registered in the PRDS register table. For more information on the audit trail records the PRU produces see *IMS DPROP Messages and Codes*.

Environment

The PRDS Registration utility is a DB2 application program that runs in batch mode using TSO attach. It can be executed in a number of environments:

- The Selector and the Receiver on the same MVS image or a local MVS image with shared DASD

The PRDSs are immediately available to the Receiver. You can invoke the PRU as a job step in the Selector JCL. When the Selector writes the trailer record to the PRDS, it automatically generates REGISTER control statements for each PRDS. Then the Selector writes the REGISTER control statements to the //EKYPRREG file. See the appropriate Administrators Guide for your propagation mode for an example of the Selector JCL.

- The Selector and the Receiver on a remote MVS image

  The Selector automatically generates the REGISTER control statements and stores them in the //EKYPRREG file. However, you must provide a transport program to send the //EKYPRREG file to the Receiver site. The PRDS register table resides on the Receiver site. On the Receiver site, you can run the PRU the first job in the Receiver JCL or as a separate job using the PRU JCL. If you run the PRU as a separate job, you can either:

  – Use REGISTER control statements from the //EKYPRREG file in the //EKYREGIN DD statement.

  – Use the //EKYPRREG file directly in the //EKYREGIN DD statement.

  You must edit the //EKYPRREG file when it arrives at the Receiver site if you are using different naming conventions

  You can run the PRU on the Receiver site to unregister or list the PRDSs in the PRDS register table. You must supply the input statements.

  If tapes are required for the PRU you must request that the tape volumes be mounted. Make the request through the user attribute data set (UADS) entry for the TSO userid on the job card used to run the PRU, or through RACF. See*TSO Extensions Version 2: Administration*.

JCL Requirements

This topic describes the following JCL:

- EKYUPRUP, the sample JCL to execute the PRDS Registration utility
- EKYUPRUJ, the sample JCL that calls EKYUPRUP

EKYUPRUP: Sample JCL to Execute the PRU

Figure 244 on page 338 is the sample JCL for the EKYUPRUP procedure which executes the PRU. Before you invoke the procedure, replace all occurrences of *1111111* with the high-level qualifiers for the IMS DPROP RESLIB.

```
//*-------------------------------------------------------------------*
//* PROCEDURE EKYUPRUP TO EXECUTE THE PRDS REGISTRATION UTILITY (PRU) *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                               *
//**        LICENSED MATERIALS - PROPERTY OF IBM                   *
//**                                                               *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                 *
//**                                                               *
//**                                                               *
//**        SEE COPYRIGHT INSTRUCTIONS                             *
//**                                                               *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S STANDARDS :    *
//*  ------------------------------------------------------------     *
//*                                                                   *
//*  1111111   - DPROP HLQ(S)                                         *
//*                                                                   *
//*-------------------------------------------------------------------*
//*
//EKYUPRUP PROC EKYPREF=1111111
//*
//*-------------------------------------------------------------------*
//*
//RUN     EXEC PGM=IKJEFT01,DYNAMNBR=20
//EKYRESLB DD  DISP=SHR,
//         DSN=&EKYPREF..EKYRESLB
//*-------------------------------------------------------------------*
//*---  DATASETS REQUIRED SPECIFICALLY BY THE PRU
//*-------------------------------------------------------------------*
//REGLIST  DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//*-------------------------------------------------------------------*
//*---  DATASETS REQUIRED BY DPROP MODULES
//*-------------------------------------------------------------------*
//EKYPRINT DD  SYSOUT=*
//EKYLOG   DD  SYSOUT=*
//*-------------------------------------------------------------------*
//*---  DATASETS REQUIRED BY IDCAMS
//*-------------------------------------------------------------------*
//SYSIN    DD  DSN=&&SYSIN,UNIT=SYSDA,SPACE=(TRK,(1)),
//         DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),DISP=(NEW,DELETE)
//SYSPRINT DD  DSN=&&SYSPRINT,UNIT=SYSDA,SPACE=(629,(20,10)),
//         DCB=(LRECL=125,BLKSIZE=629,RECFM=VBA),DISP=(NEW,DELETE)
//*
//*-------------------------------------------------------------------*
```

Figure 244. Sample JCL for EKYUPRUP

The DD statements //EKYRESLIB, //EKYPRINT, and //EKYLOG are common to many IMS DPROP components. See Chapter 1 for descriptions of these statements and other common JCL.

//REGLIST
Output from the LIST control statement and from the UNREGISTER control statement, if any.

//SYSTSPRT
A print file used by the TSO TMP environment.

//SYSIN
Contains commands generated by the PRU that are input to the IDCAMS utility, which is called by the PRU.

Allocate this DD statement to a temporary data set, *not* as a SYSIN.

//SYSPRINT
Contains the print output of the IDCAMS utility called by the PRU.

Allocate this DD statement to a temporary data set, *not* as a SYSOUT file.

Sample JCL to Call the EKYUPRUP Procedure

Figure 245 on page 340 shows the sample JCL for EKYUPRUJ, which is required to execute the PRU

Before you invoke this procedure:
- Replace *JJJJJJJ* with job card information.
- Replace 1111111 with the high-level qualifiers for the IMS DPROP RESLIB.
- Replace *2222222* with the data set name of DB2 subsystem.
- Replace *3333333* with the name of the PRDS Registration utility CP plan.

```
//JJJJJJJ  JOB  *** VALID JOB CARD DETAILS ***
//*-------------------------------------------------------------------*
//*  JCL TO CALL PROCEDURE EKYUPRUP (PRDS REGISTRATION UTILITY - PRU) *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S STANDARDS :    *
//*-------------------------------------------------------------------*
//*                                                                   *
//*  JJJJJJJ   - JOB NAME                                             *
//*  1111111   - DSN OF THE //EKYPRREG DD CREATED BY THE SELECTOR.    *
//*              THIS CAN BE REPLACED WITH A DD * STATEMENT TO        *
//*              FACILITATE INLINE CONTROL STATEMENTS.                *
//*  2222222   - DB2 SUBSYSTEM NAME                                   *
//*  3333333   - NAME OF THE PRDS REGISTRATION UTILITY (PRU) PLAN     *
//*                                                                   *
//*-------------------------------------------------------------------*
//*
//*
//RUNPRU  EXEC PROC=EKYUPRUP
//*
//EKYREGIN DD  DSN=1111111,DISP=SHR
//SYSTSIN     DD   *
 DSN SYSTEM(2222222)
 RUN  CP PLAN(3333333)
  EKYP000X
 END
/*
//*
//*-------------------------------------------------------------------*
```

Figure 245. EKYUPRUJ: Sample JCL to call EKYUPRUP

//EKYREGIN
The PRU control statements.

//SYSTSIN
The input data set for TSO batch. It contains a DSN command, a RUN CP subcommand, the PRU program name that TSO processes as a TSO subcommand, and an End command.

This data set executes the PRU as a DB2 application in batch mode.

If you have more than one DB2 subsystem installed, IMS DPROP uses this data set to get the name of the correct DB2 subsystem to perform the BIND PACKAGE or FREE PACKAGE of the SQL update.

Control statements

The PRU control statements are REGISTER, UNREGISTER, and LIST. The Selector can automatically generate only the REGISTER control statement. However, you can include any or all of the control statements in the PRDS Registration utility JCL. The PRU control statements add, delete, or list PRDSs to or from the PRDS registration table. See Figure 244 on page 338 for an example of the PRU JCL.

This topic describes the syntax and parameters of the PRU control statements. The examples given in this topic for the DSN, VOLSER, UNIT, and LABEL are site specific and are intended only as a guide. Define values for these items depending on your site requirements.

REGISTER

Use the REGISTER control statement to add the following PRDS entries to the PRDS register table:

- One specific PRDS with a fully qualified data set name. This is the only type of statement that can be generated automatically by the Selector.
- All PRDSs with a particular data set name pattern.

Figure 246 and Figure 247 on page 342 show the syntax of the REGISTER control statement.

```
►►──REGISTER──DSN=dsn──┬──────────────────────────────┬──────────────────────────────────►◄
                       │┌──,──┐                       │
                       ├──,VOLSER=ser──────────────────┤
                       │      ▼                       │
                       └──,VOLSER=(──ser──────────)───┘

►►──┬──────────────────────────────────────────────┬──;──────────────────────────────────►◄
    │                  ┌──,LABEL=(─1,SL─)──────────┐│
    └──,UNIT=unit──────┼──,LABEL=nnnn──────────────┼┘
                       └──,LABEL=(─nnnn,xxx─)──────┘
```

Figure 246. The REGISTER Statement with a Fully Qualified Data Set Name

DSN= *dsn*

The fully qualified data set name of the PRDS. If the PRDS is cataloged, do not specify values for any other parameters in the REGISTER control statement.

A data set name can be up to 44 alphanumeric characters and for ease of reference can contain the PRDS name. For example, if the PRDS name is `GROUP01`, the data set name could be `DPROPTT.TT4A514.GROUP01`.

VOLSER= *ser* **|** *(ser1,ser2,...)*

The serial numbers of the volumes where the PRDS resides. You must specify the VOLSER if the PRDS is uncataloged.

ser

The Volume Serial Number. A volume serial number is 1 to 6 alphanumeric characters.

(ser1,ser2,...)
The serial numbers of different volumes.

UNIT= *unit*
The type of unit on which the PRDS resides. You must specify the UNIT if the PRDS is uncataloged.

A UNIT type can be a cartridge, a disk, or a tape. It consists of 5 alphanumeric characters.

LABEL= *nnn* | *(nnnn,xxx)*
The label of the tape on which the PRDS resides. This parameter is only required for uncataloged tape PRDSs, where the PRDS is not the first data set on a standard labeled (SL) tape.

nnnn
The relative position of the PRDS on the tape volume.

(xxx)
The label type of the tape according to MVS JCL DD LABEL parameters. Valid values for *xxx* are `SL`, `SUL`, `AL`, `AUL`, `NSL`, `NL`, `BLP` and `LTM`.

The default value for the LABEL parameter is `1,SL`. This default is inserted into the propagation request control table for all uncataloged data sets, whether on tape or DASD.

```
►►──REGISTER──┬─ENTRIES=pattern─┬──;──────────────────────────────────►◄
              └─LEVEL=pattern───┘
```

Figure 247. The REGISTER Statement using a Data Set Name Pattern

ENTRIES= *pattern*
The data set name pattern of one or more cataloged PRDSs. The PRU registers data set names that contain the data set name pattern specified on this parameter. A pattern is a character sequence.

LEVEL= *pattern*
The data set name pattern of one or more cataloged PRDSs. The PRU registers data set names that contain the character sequence specified.

Using the Data Set Name Patterns

The format of the data set name pattern on the ENTRIES and LEVEL parameters is specified in the same way as on the LISTCAT control statement for IDCAMS. (See the*MVS/DFP Access Method Services for the Integrated Catalog Facility* for information on IDCAMS.) For example, if the catalog contains the following data set names:

```
1.A.A.B
2.A.B.B
3.A.B.B.C
4.A.B.B.C.C
5.A.C.C
6.A.D
7.A.E
8.A
```

The data set names registered depend on the parameter and the character sequence specified. Examples are:

Specify	Register
ENTRIES=A.*	items 6 and 7 from the catalog list
ENTRIES=A.*.B	items 1 and 2 from the catalog list
LEVEL=A.*.B	items 1,2,3 and 4 from the catalog list
LEVEL=A	items 1, 2, 3, 4, 5, 6, 7 and 8 from the catalog list

Recommendations for Use of the REGISTER control statement

The PRU registers data sets that are considered valid by IMS DPROP. Follow these recommendations to avoid problems:

- Always catalog data sets that are used as PRDSs.
- In a single-site environment, run the PRU as an extension of the Selector. The Selector automatically creates one file of PRU REGISTER statements for each PRDS, so you avoid problems caused by data sets that are not PRDSs, data sets that are empty, or data sets of incorrect record length. The PRU does not register uncataloged data sets from the file of REGISTER statements the Selector generates.
- In a multi-site environment, register PRDSs as part of the receive process when PRDSs are transported from the Selector site to the Receiver site. Specify the full data set name when you register the data set.

  Use a GDG for the PRDSs that are received. This streamlines the process because all the data sets are selected using the base name as a search pattern on the REGISTER statement. An example is REGISTER LEVEL=IBM.DPROP.PRDS.
- Use data set name patterns with care on ENTRIES and LEVEL keywords. If the data set names are not unique, the PRU might select:
  - Files that have different attributes than a PRDS.
  - Files that have the same data set attributes as a PRDS, but are empty (contain no records). If IMS DPROP attempts to read these files to extract the requisite registration data, an abend occurs.

Examples of the REGISTER control statement

Example 1: The example in Figure 248 registers a cataloged PRDS that has a fully qualified data set name to the PRDS register table.

```
REGISTER DSN=DPROPTT.TT4A514.GROUP01
```

Figure 248. Example 1: A Cataloged PRDS

Example 2: The example in Figure 249 registers an uncataloged PRDS that has a fully qualified data set name to the PRDS register table.

```
REGISTER DSN=DPROPTT.TT4519.GROUP03,VOLSER=HBAD01,UNIT=3480,LABEL=(3,SL  )
```

Figure 249. Example 2: An Uncataloged PRDS

Example 3: The example in Figure 250 on page 344 registers cataloged PRDSs that match the specified data set name pattern to the PRDS register table.

```
REGISTER ENTRIES=DPROPTT.*.GROUP01
```

Figure 250. Example 3: Cataloged PRDSs that Match an ENTRIES Pattern

Example 4: The example in Figure 251 registers cataloged PRDSs that match the specified data set name pattern to the PRDS register table.

```
REGISTER LEVEL=DPROPTT.*.GROUP02
```

Figure 251. Example 3: Cataloged PRDSs that Match a LEVEL Pattern

UNREGISTER

Use the UNREGISTER control statement to delete the following types of requests from the PRDS register table:

- One specific propagation request data set with a fully qualified data set name
- All propagation request data sets based on a data set name pattern and a specified age

The PRU writes a list of the PRDSs that have been unregistered to the //REGLIST file.

The syntax of the UNREGISTER control statement is shown in Figure 252 and Figure 253.

```
►►──UNREGISTER──DSN=dsn──;──────────────────────────────────────────►◄
```

Figure 252. The UNREGISTER Statement using a Fully Qualified Data Set Name

DSN= *dsn*
The fully qualified data set name of the PRDS. The PRDS is unregistered whether it is cataloged or uncataloged.

```
►►──UNREGISTER──┬─ENTRIES=pattern──┬──,──OLDERTHAN=──┬─ANY─┬──┬──────────────┬──;──►◄
                │                  │                 └─nnn─┘  └──,──CAT=EVEN──┘
                ├─LEVEL=pattern────┤
                └─ALL──────────────┘
```

Figure 253. The UNREGISTER Statement using a Data Set Name Pattern

When using the UNREGISTER control statement with the DSN pattern, the PRU checks for each PRDS that all Receivers that can process the PRDS have done so. If any Receiver has not processed the PRDS, then that PRDS is not unregistered. When you use UNREGISTER with a fully qualified DSN, the PRU does not perform this check.

ENTRIES= *pattern*
The data set name pattern of one or more cataloged or uncataloged PRDSs.

The data set name pattern specified on this parameter deletes data set names that contain the pattern. A pattern is a character sequence.

LEVEL= *pattern*
The data set name pattern of one or more cataloged or uncataloged PRDSs. The data set name pattern specified on this parameter deletes data set names that contain the pattern.

Specify the data set name pattern on the ENTRIES and LEVEL parameters the same way as on the LISTCAT control statement for IDCAMS. See the*MVS/DFP Access Method Services for the Integrated Catalog Facility* for information on IDCAMS. See “Using the Data Set Name Patterns” on page 342 for examples of data set name patterns.

ALL
Deletes all PRDSs within the age period specified on the OLDERTHAN parameter.

OLDERTHAN=ANY | *nnn*
The age of the PRDSs to be unregistered from the PRDS register table.

ANY
Deletes PRDSs of any age, including those registered today.

nnn
Deletes a PRDS according to the number of days that have elapsed between the registration of the PRDS (not the creation of the PRDS) and the current date. Valid values for this parameter are in the range 0 to 999. For example:

```
OLDERTHAN=0
```

deletes all registrations performed before today.

```
OLDERTHAN=1
```

deletes all registrations performed before yesterday.

CAT=EVEN
Specifies that PRDSs are unregistered even if they are currently cataloged in the MVS catalog. By default, this parameter is not specified, and the PRU deletes only PRDSs that are not cataloged in the MVS catalog.

Examples of the UNREGISTER control statement

Example 1: The example in Figure 254 deletes a PRDS that has a fully qualified data set name from the PRDS register table, whether it is cataloged or uncataloged.

```
UNREGISTER DSN=DPROPTT.TT451B.GROUP01
```

Figure 254. Example 1: Delete PRDS with Fully Qualified Data Set Name

Example 2: The example in Figure 255 on page 346 deletes cataloged PRDSs that match the specified data set name pattern from the PRDS register table, regardless of when they are registered.

```
UNREGISTER ENTRIES=DPROPTT.*.GROUP01,OLDERTHAN=ANY
```

Figure 255. Example 2: Delete PRDSs that Match a Data Set Name Pattern

Example 3: The example in Figure 256 deletes PRDSs from the PRDS register table that match the specified data set name pattern, that were registered before today, whether they are cataloged or uncataloged.

```
UNREGISTER LEVEL=DPROPTT.*.GROUP04,OLDERTHAN=0,CAT=EVEN
```

Figure 256. Example 3: Delete PRDSs that Match a Data Set Name Pattern

Example 4: The example in Figure 257 deletes all PRDSs from the PRDS register table regardless of when they were registered, or whether they are cataloged or uncataloged.

```
UNREGISTER ALL,OLDERTHAN=ANY,CAT=EVEN
```

Figure 257. Example 4: Delete all PRDSs

LIST

Use the LIST control statement to display:

- The information in a row of the PRDS register table for a fully qualified data set name, or information in all the rows that match a data set name pattern
- The age of a registered PRDS
- Whether the PRDS is cataloged or uncataloged
- Whether or not a PRDS is a IMS DPROP PRDS

The PRU writes information from the LIST statement to the //REGLIST file.

Figure 258 and Figure 259 on page 347 show the syntax of the LIST control statement.

```
►►──LIST──DSN=dsn──;──────────────────────────────────────────────────►◄
```

Figure 258. The LIST Statement using a Fully Qualified Data Set Name

DSN= *dsn*
The fully qualified data set name of the PRDS.

```
►►──LIST──┬─ENTRIES=pattern─┬──┬──────────────────────────┬──;──────────────────►◄
          ├─LEVEL=pattern───┤  └─,──OLDERTHAN=──┬─ANY─┬──┘
          └─ALL─────────────┘                   └─nnn─┘
```

Figure 259. The LIST Statement using a Data Set Name Pattern

ENTRIES= *pattern*
The data set name pattern of PRDSs. The data set name pattern specified on this parameter lists data set names that contain the pattern.

LEVEL= *pattern*
The data set name pattern of PRDSs. The data set name pattern specified on this parameter lists data set names that contain the pattern.

The format of the data set name pattern on the ENTRIES and LEVEL parameters is specified in the same way as on the LISTCAT control statement for IDCAMS. See*MVS/DFP Access Method Services for the Integrated Catalog Facility* for information on IDCAMS. See “Using the Data Set Name Patterns” on page 342 for examples of data set name patterns.

ALL
Lists information on all PRDSs within the age period specified on the OLDERTHAN parameter.

OLDERTHAN=ANY | *nnn*
Specifies the age of the PRDSs in the PRDS register table.

ANY
Lists information on PRDSs of any age, including those registered today.

nnn
Lists information on a PRDS according to the number of days that have elapsed between the registration of the PRDS (not the creation of the PRDS) and the current date. Valid values for this parameter are in the range 0 to 999. For example:

```
OLDERTHAN=0
```

Lists all registrations performed before today.

```
OLDERTHAN=1
```

Lists all registrations performed before yesterday.

Examples of the LIST control statement

Example 1: The example in Figure 260 lists information on a PRDS that has a fully qualified data set name.

```
LIST DSN=DPROPTT.TT45159.GROUP02
```

Figure 260. Example 1: List a PRDS that has a Fully Qualified Data Set Name

Example 2: The example in Figure 261 on page 348 lists information on PRDSs that match the specified data set name pattern.

```
LIST ENTRIES=DPROPTT.*.*
```

Figure 261. Example 2: List PRDSs that Match a Data Set Name Pattern

Example 3: The example in Figure 262 lists information on PRDSs that match the specified data set name pattern and were are registered before yesterday.

```
LIST LEVEL=DPROPTT.*.GROUP05,OLDERTHAN=1
```

Figure 262. Example 3: LIST PRDSs of Specific Age

Example 4: The example in Figure 263 lists information on all PRDSs, regardless of when they are registered.

```
LIST ALL
```

Figure 263. Example 4: LIST All PRDSs

Output Messages

Error messages for the PRU have prefixes of EKYP. For more information on PRU messages and codes, see *IMS DPROP Messages and Codes*.

Return Codes and Error Conditions

The PRU provides the following return codes:

Code **Meaning**

0 No error has occurred.

4 Warning messages are issued for one or more of the following reasons:

Subcode
Meaning

0 File problem
- No DSNs returned from catalog for DSN pattern on REGISTER statement
- Deallocate problem on REGISTER statement
- Unexpected return code from IDCAMS for REGISTER statement

4 Cannot register because PRDS already exists, or cannot unregister or list because PRDS is not registered.

8 No PRDS found to satisfy the OLDERTHAN operand.

12 Cannot unregister, all PRDSs returned are cataloged.

16 Allocation of a PRDS failed.

32 The PRU is unable to read the first record of a PRDS.

64 A data set is not a PRDS data set.

8 A severe error was detected.
- Internal error

- Unexpected SQL return code
- Syntax error on control statement

Chapter 13. CDCDS Utilities

A Changed Data Capture data set (CDCDS) contains IMS log records required for propagation. It can be used as input to the Selector in place of the IMS System Log data set (SLDS) to reduce Selector run time.

The IMS Archive utility can produce a CDCDS during the archiving of on-line log data sets (OLDS) to SLDS. To produce a CDCDS, use the COPY function of the IMS Archive utility (DFSUARC0) and a set of RECORD statements that select the IMS log record types required for propagation, as described in the appropriate Administrators Guide for your propagation mode.

A CDCDS must be registered to DBRC when it is created. You register a CDCDS by using DBRC commands to create two records in the RECON data sets, one for the CDCDS and one for the SLDS it relates to. Then the Selector can determine that a CDCDS exists and use it in preference to the corresponding SLDS.

The following topics provide additional information:
- “CDCDS Registration Utility ”
- “CDCDS Unregistration Utility ” on page 355

The JCL described in this topic is:
- EKYUCRUP to execute the CRU, described on page 352
- EKYUCDUP to execute the CDU, described on page 356

CDCDS Registration Utility

The CDCDS Registration utility (CRU) registers new CDCDSs to DBRC. You can execute the CRU either as part of the IMS Archive utility JCL, or independently.

Overview

Use the following CDCDS commands to register each CDCDS after it is created:

NOTIFY.PRILOG
To register the CDCDS

NOTIFY.SECLOG
To register the associated SLDS

The CDCDS automatically generates and executes these commands, creating two entries in DBRC which are stored in the RECON data sets. The Selector uses the timestamp of these entries to locate the CDCDS entry. When the Selector finds the entry it dynamically allocates the CDCDS.

If you have multiple IMS systems which share RECONs, it is possible for a CDCDS registration to fail because of a *duplicate key* condition. This only occurs when two or more IMS systems open an OLDS at exactly the same time. The corresponding CDCDSs have the same start time. If the CRU runs after the first of the CDCDSs, it fails, and that CDCDS is not registered.

This has no effect on the Selector, other than to force it to use the SLDSs. If this happens, the Selector might take longer to run because of the additional IMS log records it must read in order to obtain the propagation records. In a multi-site environment, the CRU runs on the Selector site only.

Input and Output

The input to the CRU is the value the user gives in the PARM parameter of the EXEC statement in the sample JCL for the CRU. See Figure 264 on page 353 for the sample JCL. This contains the TIMESTAMP that is associated with the two DBRC records. The format of the timestamp is:

```
YYDDDHHMMSSN
```

where:

YY	Represents the two digits of the year
DDD	Represents the number of the day
HH	Represents the hour (include zeros)
MM	Represents the minutes (include zeros)
SS	Represents the seconds (include zeros)
N	Represents tenths of a second

For example:

```
931211148231
```

This is the standard timestamp format as used by DBRC.

Output from the CRU consists of:

- The DBRC commands NOTIFY.PRILOG and NOTIFY.SECLOG which are used as input to DBRC.
- The SELPRINT file, which contains messages and error information.

EKYUCRUP—JCL to Run the CRU

Figure 264 on page 353 is the sample JCL for the EKYUCRUP procedure, which executes the CRU.

```
//JJJJJJJ JOB  ***  VALID JOB CARD DETAILS  ***
//*********************************************************************
//**                                                                 */
//**        LICENSED MATERIALS - PROPERTY OF IBM                     */
//**                                                                 */
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   */
//**                                                                 */
//**                                                                 */
//**        SEE COPYRIGHT INSTRUCTIONS                               */
//**                                                                 */
//*********************************************************************
//*********************************************************************
//*  DESCRIPTION:                                                     *
//*                                                                   *
//*  THIS UTILITY GENERATES A NOTIFY.PRILOG COMMAND FOR THE CDCDS     *
//*  AND A NOTIFY.SECLOG COMMAND FOE THE ASSOCIATED SLDS. THE NAME    *
//*  OF THE SLDS AND THE CDCDS ARE SPECIFIED USING THE CORRESPONDING  *
//*  DD STATEMENTS. THE TIMESTAMP TO BE ASSIGNED TO THE ENTRIES IS    *
//*  SPECIFIED VIA THE  PARM STRING.                                  *
//*                                                                   *
//*  HAVING GENERATED THESE CONTROL STATEMENTS, THE UTILITY THEN      *
//*  EXECUTED THEM BY MAKING A CALL TO DSPURX00, THE DBRC INTERFACE.  *
//*                                                                   *
//*  THIS UTILITY SHOULD BE INVOKED AS PART OF THE IMS ARCHIVE JCL    *
//*                                                                   *
//*-------------------------------------------------------------------*
//* CHANGE THE FOLLOWING TO COMPLY WITH YOUR INSTALLATION STANDARDS:- *
//*                                                                   *
//*       1111111  - THE DPROP RESLIB HLQ(S)                          *
//*       2222222  - THE DATASET NAME OF THE SLDS                     *
//*       3333333  - THE DATASET NAME OF THE ASSOCIATED CDCDS         *
//*       4444444  - THE TIMESTAMP IN THE FORM 'YYDDDHHMMSST' THAT IS *
//*                  TO BE ASSIGNED TO THE DBRC ENTRIES.              *
//*                                                                   *
//*-------------------------------------------------------------------*
//*       DEPENDENCIES : NONE
//*-------------------------------------------------------------------*
//*
//EKYUCRU  PROC EKYPREF=1111111,
                SLDSNAME=2222222,
                CDCDSNAM=3333333,
                PRM=4444444
//EKYB800X EXEC PGM=EKYB800X,PARM=&PRM
//EKYRESLB DD   DSN=&EKYPREF..EKYRESLB,DISP=SHR
//SYSPRINT DD   DSN=&&DBRCOUT,DISP=(,DELETE,DELETE),UNIT=SYSDA,
//              SPACE=(TRK,10)
//SELPRINT DD   SYSOUT=*
//SLDS     DD   DSN=&SLDSNAME,DISP=SHR
//CDCDS    DD   DSN=&CDCDSNAM,DISP=SHR
//SYSIN    DD DSN=&&EKYCDCIN,DISP=(,DELETE,DELETE),UNIT=SYSDA,
//         SPACE=(TRK,5),DCB=RECFM=FB,LRECL=80,BLKSIZE=800)
```

Figure 264. Sample JCL for EKYUCRUP

The DD statement //EKYRESLB is common to many IMS DPROP components. This statement and other common JCL is described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The remaining DD statements are:

EXEC

Executes the CRU and passes the required timestamp to the program as a parameter. The timestamp must be in standard DBRC format and enclosed in apostrophes. The CRU checks the format and rejects it if invalid.

//SLDS
Names the SLDS data set produced by the IMS Archive utility during the same run that produced the CDCDS. The CRU does not open this data set, but uses it to supply the data set name used in the DBRC commands.

//CDCDS
Names the CDCDS data set produced by the IMS Archive utility produced during the same run that produced the SLDS. The CRU does not open the data set, but uses it to supply the data set name used in the DBRC commands.

//SELPRINT
Names the data set to which all output messages are written. DBRC writes its messages to the SYSPRINT data set, and the CRU copies the DBRC messages to the SELPRINT data set. The CRU writes its own messages directly to SELPRINT.

//SYSPRINT
Used for reports produced by DBRC. The CRU checks the DBRC reports for error messages, in particular for a *duplicate key* error. A duplicate key error results when an attempt is made to create a DBRC record with a key that already exists. This happens when the start time used in the NOTIFY.PRILOG command to register the first CDCDS of an active online IMS system is the same as the start time of the active online IMS system. When the CRU detects this condition, it increases the start time used for the NOTIFY.PRILOG command by one tenth of a second and tries again. It repeats this up to a maximum of 100 times, then the program fails with a return code of 12. For this reason the timestamp of the CDCDS record can differ from that of the associated SLDS.

Because this data set is used as input to the CRU, it must *not* be allocated as SYSOUT.

//SYSIN
Used to hold the DBRC commands prior to their execution. Figure 265 shows the commands.

```
NOTIFY.PRILOG RLDS DSN(CDCDS-name) STARTTIME(parm-timestamp)
SSID(ACDCSSID) VOLUME(EKY000) IMS
NOTIFY.SECLOG RLDS DSN(SLDS-name) STARTTIME(parm-timestamp)
SSID(ACDCSSID) VOLUME(EKY000) IMS
```

Figure 265. CRU DBRC Commands

The values for the DSN parameters are the data set names associated with the CDCDS and SLDS DD statements respectively. The value for the STARTTIME parameter is the timestamp specified in the PARM parameter.

The VOLUME keyword and value have to be included to satisfy DBRCs syntax checking. They are not used by the Selector, which locates its input data set through the MVS catalog structure. Therefore, all SLDS and CDCDS files *must* be catalogued.

Output Messages

Output message prefixes are EKYB8 for CRU messages, and DSP for DBRC utility messages. For more information on CRU messages and codes, see IMS DPROP Messages and Codes.

Return Codes and Error Conditions

The CRU provides the following return codes:

Code	Meaning
0	Registration completed successfully.
12	Registration completed with errors.

CDCDS Unregistration Utility

The CDCDS Unregistration utility (CDU) deletes CDCDS entries in DBRCs that are no longer required. These are usually CDCDSs that have been processed successfully by the Selector.

Overview

The DBRC CDCDS entries have a stop time of 0 and are therefore protected from deletion by routine DBRC cleanup operations. It is up to you to perform a cleanup operation using the CDCDS Unregistration utility.

The CDCDS Unregistration utility uses the DBRC GENJCL.USER command to delete CDCDS entries. The CDCDS Unregistration utility uses the current MVS date and time, and a user-supplied *older than nn days* parameter to compute the range of dates within which to delete the CDCDS records. The CDCDS Unregistration utility then uses this information to build the required DBRC command.

Ways to Delete CDCDS Entries

The CDCDS Unregistration utility deletes only the DBRC records that relate to CDCDSs. It does not delete the actual data sets. Ways you can delete CDCDS entries are:

- Delete the unwanted CDCDSs manually.
- Use an generation data group (GDG) for the CDCDSs and specify the SCRATCH option and a suitable LIMIT value when building the GDG index. The limit value should allow at least enough time for the CDCDS to be processed successfully by the Selector and, if possible, enough time to ensure that the resultant PRDSs are successfully processed by the Receiver.
- Modify the sample JCL in Figure 269 on page 359 to include a DD statement for the CDCDS with a DISP=(OLD,DELETE). The format of the statement should be as shown in Figure 266. It can be inserted in the sample anywhere between the EXEC statement and the //SYSIN DD statement.

```
//CDCDS  DD  DSN=%LOGDSN,DISP=(OLD,DELETE)
```

Figure 266. Sample DD Statement to Delete the CDCDS

The example in Figure 266 deletes the actual CDCDS that corresponds to the DBRC entry selected by the CDCDS Unregistration utility. UNIT and VOLUME parameters are not included because CDCDSs *must* be cataloged.

Input and Output

The input to the CDCDS Unregistration utility is the value you give in the PARM parameter of the EXEC statement in the sample JCL for the CDCDS Unregistration

utility. See Figure 267 on page 357 for the sample JCL. Each CDCDS record which is older than the number of days specified is deleted.

PARM=*nn* deletes CDCDS records older than *nn* days from the current day. Specify one of the following values in the *nn* field:

00 Deletes all CDCDS records

01-99 Deletes CDCDS records older than the specified number of days

For example, if you specify the value *03*, CDCDS records three days older than the current date and time is deleted.

The output from the CDCDS Unregistration utility is:

- The DBRC GENJCL.USER command (see Figure 268 on page 358) which uses the predefined skeletal member, EKYCDDEL (see Figure 269 on page 359), to generate executable JCL.

  The generated JCL deletes each CDCDS record older than nn days from the RECON data sets. The sample JCL writes the generated JCL to the internal reader, resulting in the immediate execution of the JCL. Alternatively, you can specify the name of a data set to hold the generated JCL so it can be submitted later.
- The SELPRINT file, which contains messages, including any errors.

Environment

In a multi-site environment the CDCDS Unregistration utility runs on the Selector site only, and there are no implications for the Receiver site.

EKYUCDUP—JCL to Run the CDU

Figure 267 on page 357 is the sample JCL for the EKYUCDUP procedure, which is used to execute the CDCDS Unregistration utility.

```
//*             DPROP VERSION 2 RELEASE 1                          *
//*       PROCEDURE EKYUCDEP TO DELETE CDCDS ENTRIES FROM DBRC       *
//*******************************************************************
//**                                                               */
//**        LICENSED MATERIALS - PROPERTY OF IBM                   */
//**                                                               */
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                 */
//**                                                               */
//**        SEE COPYRIGHT INSTRUCTIONS                             */
//**                                                               */
//*******************************************************************
//*******************************************************************
//*  DESCRIPTION:                                                   *
//*                                                                 *
//*  STEP1 - EKYB500X BUILDS THE FOLLOWING DBRC COMMAND IN THE      *
//*          TEMPORARY DATASET ASSIGNED TO //EKYCDCIN:-             *
//*                                                                 *
//*  GENJCL.USER MEMBER=EKYCDDEL USERKEYS((%UTIME,'USERTIME')) NOJOB *
//*                                                                 *
//*     USERTIME' WILL CONTAIN THE CURRENT DATE/TIME MINUS THE      *
//*     NUMBER OF DAYS SPECIFIED IN THE PARM= PARAMETER.            *
//*                                                                 *
//*  STEP2 - DSPURX00 (DBRC) PROCESSING                             *
//*                                                                 *
//*       EXECUTES DBRC USING THE SYSIN STATEMENT BUILT BY EKYB500X *
//*       THIS USES THE SKELETON JCL MEMBER EKYCDDEL FROM THE PDS   *
//*       NAMED IN //JCLPDS TO GENERATE JCL. *
//*                                                                 *
//*       THE RESULTANT JOB WILL DELETE FROM THE RECONS EACH        *
//*       CDCDS OLDER THAN 'USERTIME'.  THE JCL IS WRITTEN TO THE   *
//*       DATASET //JCLOUT. IF IMMEDIATE EXECUTION IS REQUIRED, THIS *
//*       SHOULD BE CHANGED TO POINT TO THE INTERNAL READER.        *
//*                                                                 *
//*       //JCLPDS AND //JCLOUT ARE STANDARD DBRC DDNAMES.          *
//*-----------------------------------------------------------------*
//*       CHANGE AS REQUIRED :
//*
//*       1111111  - THE DPROP RESLIB HLQ(S)
//*       2222222  - THE HLQ(S) OF THE IMS LIBRARIES
//*       3333333  - THE NUMBER OF DAYS PRIOR TO WHICH CDCDS SHOULD
//*                  BE DELETED (RANGE 1 TO 99)
//*-----------------------------------------------------------------*
//*       DEPENDENCIES : NONE
//*-----------------------------------------------------------------*
//EKYUCDEL PROC EKYPREF=1111111,
                IMSPREF=2222222,
                PRM=3333333
//EKYB500X EXEC PGM=EKYB500X,PARM=&PRM
//EKYRESLB DD   DSN=&EKYPREF..EKYRESLB,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SELPRINT DD   SYSOUT=*
//EKYCDCIN DD   DSN=&&EKYCDCIN,DISP=(NEW,PASS),
//         DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),
//         SPACE=(800,(10,1),RLSE)
//*
//STEP2    EXEC PGM=DSPURX00,COND=(4,LT)
//SYSPRINT DD SYSOUT=*
//***
```

Figure 267. EKYUCDEP: Sample JCL to Execute the CDCDS Unregistration Utility (Part 1 of 2)

```
//***
//* PDS CONTAINING SKELETAL JCL MEMBER EKYCDDEL
//* (SUPPLIED IN THE SEKYSKEL INSTALLATION LIBRARY)
//*
//JCLPDS   DD DSN=&IMSPREF..JCLPDS,DISP=SHR
//***
//* DESTINATION OF THE GENERATED JOB - CAN BE CHANGED
//* TO THE INTERNAL READER FOR DIRECT SUBMISSION
//*
//JCLOUT   DD DSN=&IMSPREF..JCLOUT,DISP=SHR
//SYSIN    DD DSN=&&EKYCDCIN,DISP=(OLD,DELETE)
```

Figure 267. EKYUCDEP: Sample JCL to Execute the CDCDS Unregistration Utility (Part 2 of 2)

The DD statements //EKYRESLB and //SELPRINT are common to many IMS DPROP components and are described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The remaining statements are:

//EKYCDCIN
Contains the name of the temporary data set to which the CDCDS Unregistration utility writes the DBRC command. Figure 268 shows the DBRC command.

```
//*       EKYB500X BUILDS THE FOLLOWING DBRC COMMAND IN THE
//*       TEMPORARY DATASET //EKYCDCIN
//*
//*  GENJCL.USER MEMBER=EKYCDDEL USERKEYS((%UTIME,'USERTIME')) NOJOB
```

Figure 268. CDCDS Unregistration Utility DBRC Command

USERTIME contains the current date and time minus the number of days specified in the PARM parameter.

//SYSPRINT
Contains the print output of the IDCAMS utility called by the CDCDS Unregistration utility.

Allocate this DD statement to a temporary data set and *not* as a SYSOUT file.

//JCLOUT
Contains the name of the internal reader or the name of the data set to hold the generated JCL to submit later.

//SYSIN
Contains commands generated by the CDCDS Unregistration utility for input to the IDCAMS utility, which is called by the CDCDS Unregistration utility.

Allocate this DD statement to a temporary data set and *not* as a SYSIN file.

This utility uses DBRC and therefore requires access to the DBRC RECON data sets. If you are using DBRC without dynamic allocation, you must add the following DD statements to the sample JCL.

```
//RECON1 DD DSN=recon1,DISP=SHR
//RECON2 DD DSN=recon2,DISP=SHR
//RECON3 DD DSN=recon3,DISP=SHR
```

Figure 269 on page 359 shows the sample skeleton member, EKYCDDEL.

```
//jjjjjjjjjj JOB  *** Valid JOB card details ***
//*-------------------------------------------------------------------*
//*       SKELETON JCL USED BY DBRC TO DELETE CDCDSS FROM RECONS
//*
//*-------------------------------------------------------------------*
%SELECT RLDS (ACDCSSID,(TO(%UTIME)))
//STEP1    EXEC PGM=DSPURX00
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE.LOG STARTIME(%LOGSTIM)
%ENDSEL
```

Figure 269. CDCDS Unregistration Utility Skeletal Member

Output Messages

Error messages for the CDCDS Unregistration utility have numbers preceded by EKYB5. For more information on CDCDS Unregistration utility messages and codes, refer to IMS DPROP Messages and Codes.

Return Codes and Error Conditions

The CDCDS Unregistration utility provides the following return codes:

Code	Meaning
0	No error has occurred.
4	A warning message is issued if there are no CDCDSs to delete.
12	An invalid number of days was specified on PARM parameter.

Part 6. MQ-ASYNC Components and Utilities

Chapter 14. Capture Component

The Capture component of MQ-ASYNC is implemented in an IMS data capture exit routine that uses MQSeries to continuously transmit the IMS database changes to the Apply Program occurrences. The name of this IMS data capture exit routine is EKYMQCAP.

EKYMQCAP executes in the same environment, IMS batch regions, and IMS online regions as the updating IMS application programs; however, its execution is transparent to the updating IMS application programs. Therefore, the logic of the updating IMS application programs is not affected by EKYMQCAP.

The IMS database administrator uses the EXIT=EKYMQCAP keywords of the DBDGEN to identify which physical IMS databases (and optionally which segment types of these databases) to capture with the EKYMQCAP exit routine. EKYMQCAP processes changes of these databases and segment types according to control statements provided in the //EKYTRANS and //EKYIN files; EKYMQCAP is not aware of changes to other databases and segment types.

The MQSeries activities of EKYMQCAP and the IMS database update activities of the application programs are coordinated with a two-phase commit protocol. For IMS batch jobsteps that updated propagated IMS databases, MQ-ASYNC requires that IMS use RRS (Resource Recovery Services of the operating system) for the commit coordination.

To allow an IMS batch region or IMS online dependent region to propagate data both synchronously and with MQ-ASYNC, the synchronous DPROP system and the MQ-ASYNC Capture system must be defined during DPROP installation within the same IMS DPROP environment.

The following topics provide additional information:

EKYMQCAP Error Handling

When an IMS application program updates a database defined with EXIT=EKYMQCAP, the MQSeries Queue Managers must be active. Also, the MQSeries queues used by EKYMQCAP must be defined and ready to use. If not, EKYMQCAP will rollback the updating IMS application program (so that IMS input messages are not deleted) and will then abend.

When MQSeries or its required resources are not available and cannot be made readily available, and when propagation has a lower priority than other IMS database update applications, the operator can
- use the CUT utility to request an emergency stop of the Capture activities of an entire MQ-ASYNC Capture system, or
- provide a PROP OFF control statement in the //EKYIN file of a particular jobstep that updates IMS source databases.

This stops Capture activities and allows the updating IMS application programs to proceed without rollbacks and without abends. However, this results in data

inconsistencies between IMS source databases and DB2 target tables. You must eventually resynchronize the DB2 target tables with the IMS source; for large databases, this can be a lengthy process.

JCL for the Capture Component

EKYMQCAP accesses various files in the IMS batch and dependent online regions, where it gets control as a result of IMS database updates of the IMS application programs.

//EXEC Statement for IMS Batch

For IMS batch jobsteps the update IMS databases propagated by MQ-ASYNC, IMS DPROP requires that IMS use RRS (Resource Recovery Services of the operating system) for commit coordination. Therefore, on the EXEC statement that invokes the IMSBATCH or DBBBATCH JCL procedure, provide a RRS=Y keyword.

//EKYMQST DD Statement (Status file)

The MQ-ASYNC status file records whether an MQ-ASYNC Capture system is active or inactive. The EKYMQCAP exit routine periodically accesses the status file to determine whether the CUT utility has been used for an emergency stop, or for a reactivation of MQ-ASYNC Capture activities.

Each MQ-ASYNC Capture system has its own status file. It must be a PDS member. The PDS data set name and PDS member name are specified during installation of IMS DPROP. When the propagated IMS databases are updated in a cross-system sharing Sysplex environment, the MQ-ASYNC status file must also be shared across the Sysplex.

The MQ-ASYNC status file must be allocated in one of two ways:

- through a JCL DD statement, or
- through a default dynamic allocation to each IMS batch region and IMS dependent online region that updates IMS databases defined with the EKYMQCAP IMS data capture exit routine.

An IMS DPROP installation option specifies whether IMS DPROP will perform a dynamic allocation of a default MQ-ASYNC status file. By using the dynamic allocation of a default status file, you do not have to modify JCL for many IMS batch regions and IMS online regions.

You can override dynamic allocation by providing an //EKYMQST DD statement in the JCL of the batch and online dependent regions that update IMS databases defined with the EKYMQCAP IMS data capture exit routine. The DSN= keyword of the //EKYMQST DD statement must identify a combination of a PDS name and PDS member name that were associated during IMS DPROP installation with one of the IMS DPROP Capture systems.

The MQ-ASYNC status file must be a PDS member; it has the following characteristics, which are specified internally by IMS DPROP:

- RECFM=F or RECFM=FB
- LRECL=120

The //EKYTRANS file

The //EKYTRANS transmission specification file contains the DPROP control statements that provide the most important definitions of the MQ-ASYNC Capture component.

The //EKYTRANS transmission specification file is allocated in one of two ways:

- through a JCL DD statement, or
- through a default dynamic allocation to each IMS batch region and to each IMS dependent online region that updates IMS databases defined with the EKYMQCAP data capture exit routine.

Typically, all updating batch and online regions of the same MQ-ASYNC Capture system will use the same //EKYTRANS file. However, you can also use different //EKYTRANS data sets for different updating batch and online dependent regions. This can be useful in test environments when different tests use different propagation definitions.

By default, based on a PDS data set name and PDS member definitions provided during IMS DPROP installation for each MQ-ASYNC Capture system, the //EKYTRANS data set is allocated dynamically by IMS DPROP. With dynamic allocation, you do not have to modify JCL for many IMS batch regions and IMS online regions.

You can override this default allocation by providing an //EKYTRANS DD statement in the JCL of the batch regions and online dependent regions that update IMS databases defined with the EKYMQCAP IMS data capture exit routine.

//EKYTRANS must be a sequential file, or preferably a PDS member, and must have these characteristics:

- RECFM=F or FB
- LRECL=80

The EKYMQCAP exit routine periodically checks whether the transmission file specifications have changed. This does not apply to //EKYTRANS DD statements defined as sysin.

IMS DPROP attempts to exploit VLF to efficiently check for changes to the transmission specifications. This exploitation of VLF is possible only when

- the //EKYTRANS DD statement describes a member of the PDS that was defined during DPROP installation as the default transmission specification PDS of the MQ-ASYNC Capture system, and
- the required VLF PDS class was defined to z/OS.

  Ensure that the EKYTRANS file is included in the regular backup procedures of your installation.

  EKYTRANS contains the following mandatory control statements:
  - 1 QMANAGER control statement
  - 1-*n* PRSTREAM control statements
  - 1-*n* DB control statements

It also contains 1-*n* optional FIELD control statements.

Two examples of an //EKYTRANS DD statement follow:

```
//EKYTRANS  DD   DSN=IMSDPROP.EKYTRANS(TRANS01),DISP=SHR
```

Figure 270. First Example of an //EKYTRANS DD Statement

```
//EKYTRANS   DD   *
QMANAGER NAME=MQ01 ;
PRSTREAM NAME=STREAM1, QUEUE=MQ.LONDON.QUEUE1 ;
DB DBD=CUSTOMER ;
DB DBD=ACCOUNT  ;
 ...
 ...
/*
```

Figure 271. Second Example of a //EKYTRANS DD Statement

The //EKYIN DD Statement

The optional //EKYIN file is a secondary source of control information for the Capture component of MQ-ASYNC. The control statements of the //EKYIN file can be used to override default options of the Capture component.

The default options are appropriate in most cases; you will rarely need to provide control statements in the //EKYIN file. However, when the default options must be overridden, you must:

- Provide an //EKYIN DD JCL statement in the JCL of those IMS batch and dependent online regions that update IMS databases propagated by MQ-ASYNC.
- Provide the overriding control statements in the //EKYIN file.

For the Capture component of MQ-ASYNC, the following control statements can be provided in the //EKYIN file:

- PROP LOAD (to propagate segments inserted in database load or reload mode)
- PROP OFF (to turn off propagation in this IMS region)
- TRACE
- TRDEST

For a more detailed description of these control statements, refer to Chapter 2, "//EKYIN Data Set control statements ," on page 11.

Note: For MQ-ASYNC, you do not need to execute the SCU with ALLOWPROPOFF to use PROP OFF, because execution of the SCU with ALLOWPROPOFF is not supported for MQ-ASYNC.

Other JCL Statements for The Capture Component of MQ-ASYNC

The following JCL DD statements are required by the Capture component of MQ-ASYNC in the JCL of the updating batch regions and dependent online regions:

- //STEPLIB and //JOBLIB

  These are required if the load module libraries containing IMS DPROP load modules and user exits are not in the linklist. MQSeries load module libraries must also be accessible via linklist, joblib, or steplib.
- //EKYRESLB

  This is required only if the IMS DPROP default dynamic allocation or a user Allocation exit routine needs to be overridden.

- //EKYLEOPT

  This is required only if the user provides exit routines written in a high-level language that LE/370 supports.
- //EKYPRINT

  This DD statement is optional. With it, you can override the default dynamic allocation of //EKYPRINT as a SYSOUT(*) data set.
- //EKYWTO

  This DD statement is optional.
- //EKYSNAP

  This DD statement is optional. It is used to write OS/390® snaps of the entire OS/390 task when IMS DPROP encounters severe problems.

  When an IMS DPROP trace has been activated for a batch or online region with a TRACE control statement in //EKYIN, include the following JCL DD statements in the JCL of the updating batch or dependent online region:
- //EKYTRACE
- //EKYLOG

For a more detailed description of these JCL statements, refer to Chapter 1, “Common JCL for IMS DPROP Components,” on page 3.

JCL Statement Requirements for the Use of MQSeries

With MQ-ASYNC, jobsteps that update propagated IMS databases use MQSeries. If these jobsteps do not already use MQSeries, the JCL of these jobsteps might need to be modified to the requirements of MQSeries.

The differences between JCL requirements for IMS batch regions and IMS online dependent regions is described below.

MQSeries JCL Requirements for IMS Batch Regions: The JCL of the IMS batch regions updating IMS databases propagated by MQ-ASYNC must be able to access the load module libraries containing the MQSeries load modules. If these MQSeries load module libraries are not in the Linklist, then they must be included in the concatenation of the //JOBLIB or //STEPLIB DD statement.

MQSeries JCL Requirements for IMS Online Dependent Regions: The JCL of the IMS online dependent regions (for example, BMP Regions, Message Processing Regions, Fast Path Regions) that update IMS databases propagated by MQ-ASYNC has the following MQSeries related requirements:

- //JOBLIB or //STEPLIB

  The load module libraries containing the MQSeries load modules must be accessible to these IMS online dependent regions. If these MQSeries load module libraries are not in the Linklist, then they must be included in the concatenation of the //JOBLIB or //STEPLIB DD statement. For MQSeries Version 2 Release 1, these MQSeries load module libraries are *xxxxx*.SCSQAUTH and *xxxxx*.SCSQANL*y* (where *y* is the language letter).
- //DFSESL

  If the //JOBLIB or //STEPLIB concatenation is not APF-authorized, you should also include the MQseries load module libraries *xxxxx*.SCSQUAUTH and *xxxxxx*.SCSQANL*y* (where *y* is the language letter) in the concatenation of the //DFSESL DD statement.
- SSM= keyword and subsystem member (SSM) in the IMS.PROCLIB library.

IMS code executing in the IMS online dependent region must have access to a subsystem member of IMS.PROCLIB that defines to IMS the MQSeries subsystems that will be used in this IMS online dependent region.

Recommendation: The Region Error Option (REO) in this SSM member should be set to R. This allows IMS DPROP to regain control if its MQSeries operations fail.

By default, when no SSM= keyword is provided on the invocation of the JCL procedure for the IMS online dependent region, the IMS online dependent region will access (via the //PROCLIB DD statement) the same SSM member of IMS.PROCLIB as the SSM member used by the IMS control region. This default does not require JCL modifications (assuming that the JCL of the IMS online dependent region has a //PROCLIB DD statement pointing to the IMS.PROCLIB of the IMS control region).

As an option, the IMS online dependent region could use other SSM members as the IMS control region. In this case, you must identify the SSM member on the SSM= keyword that invokes the JCL procedure of the IMS online dependent region. If this SSM member does not yet exist in the IMS.PROCLIB, you must create it.

Control statements in the //EKYTRANS file

The //EKYTRANS transmission specification file contains the IMS DPROP control statements that provide critical definitions of the MQ-ASYNC Capture component. These control statements include:

- Definitions of the propagation data streams and
- Definitions of data (which physical IMS DBD and which segment types) that will be propagated by each propagation data stream.

The transmission specification file contains the following control statements:

1 QMANAGER control statement, mandatory
1-*n* PRSTREAM control statements, mandatory
1-*m* DB control statements with an optional SEG= or SEGEXCL= keyword
1-*n* KEYRANGE control statements, optional
1-*p* FIELD control statements, optional

Even though not required by MQ-ASYNC, it is recommended that you provide the QMANAGER control statement at the beginning of the //EKYTRANS file.

As with most IMS DPROP control statements, the //EKYTRANS control statements can include comments. A comment must start with the two characters /* and must end with the two characters */.

QMANAGER control statement

The QMANAGER control statement is mandatory. You must provide one, and only one, in the //EKYTRANS file.

The QMANAGER control statement identifies which MQSeries Queue Manager is used to send the changed data to the Apply component.

The name of the MQSeries Queue Manager should not be changed in an //EKYTRANS file while the file is being allocated to an IMS region that updates the propagated IMS databases.

Syntax of QMANAGER control statement

```
►►──QMANAGERNAME=name_of_mqseries_queue_manager──;──────────────────────────►◄
```

NAME= *name_of_mqseries_queue_manager*
identifies the name of the MQSeries Queue Manager that will be used by the Capture component of MQ-ASYNC to send the changed data to the Apply Component.

PRSTREAM control statements

The PRSTREAM control statement is mandatory. //EKYTRANS must contain one or more PRSTREAM control statements. Each PRSTREAM defines one stream of propagated data flowing from the Capture component of MQ-ASYNC to the Apply programs.

Each PRSTREAM control statement is followed by one or more DB control statements. The captured data identified by the DB control statements that follow a PRSTREAM (and that are located before the next PRSTREAM) are transmitted and propagated via this PRSTREAM.

Note: If it is important for the applications accessing the target DB2 tables that IMS changes of different IMS segment types be applied to DB2 in the same sequence as the original IMS DB updates, include the changed data of these IMS segment types in the same PRSTREAM. For example, the propagation of the entity segment type and of all extension segment types of a mapping case 2 PR should be transmitted as part of the same PRSTREAM.

Syntax of PRSTREAM control statement

```
►►──PRSTREAMNAME=name_of_PRSTREAM,QUEUE=name_of_mqseries_queue──────────────►

                                 ┌─MVS─┐
►─,MAXMSGSIZE=n,COMPRESS=──────┤     ├──;──────────────────────────────────────►◄
                                 └─NO──┘
```

NAME= *name_of_PRSTREAM.*
Assigns a name to the propagation data stream. Valid values for *name_of_PRSTREAM* include 1-8, alphanumeric, or special characters $, #, @. The first character cannot be numeric.

Within the transmission specification file, the names of the PRSTREAMs must be unique; in other words, each PRSTREAM control statement must have a different *name_of_PRSTREAM*.

QUEUE=
Identifies the name of a MQSeries queue (or the name of an MQSeries queue alias) that has been reserved for use by MQ-ASYNC. The IMS Data Capture exit routine writes the propagation data stream into the MQSeries queue identified on the QUEUE= keyword (the Capture component sends the propagation data stream via the MQSeries queue identified on the QUEUE= keyword).

When the Capture component of MQ-ASYNC is executing, the MQSeries queue or alias identified here must exist and be ready for use.

More than one PRSTREAM control statement can specify the same MQSeries queue name. If multiple PRSTREAMs specify the same queue name, all of the data of all the specified PRSTREAMs flow in FIFO (first in, first out) sequence to the Apply program.

MAXMSGSIZE=
Specifies the maximum size of MQSeries message to be built by the IMS Data Capture exit routine. The Capture component includes as many database updates in the MQSeries message as possible. The default value for maximum message size is 64512.

Note: If more than one PRSTREAM specifies the same MQSeries Queue name, each will use the same maximum message size. This size is the highest specified value in all PRSTREAMs having the same queue name.

COMPRESS=
MVS is the default value. If COMPRESS=MVS, the Capture component performs MVS compression before sending the build message to MQSeries.

Specify COMPRESS=NO if the target site does not support MVS compression and decompression.

DB control statements

The DB control statement is mandatory. It identifies physical IMS databases whose captured data is transmitted to an Apply program as part of the propagation data stream identified in the previous PRSTREAM control statement.

Note: Changes to an IMS database or to an IMS segment type are captured by IMS and provided to EKYMQCAP only if the DBD of the physical IMS database specifies the appropriate EXIT=EKYMQCAP keyword.

By default, if neither a SEG= nor a SEGEXCL= keyword is provided on the DB control statement, the captured data of *all* segment types of the named physical database is transmitted as part of the propagation data stream.

With a SEG= keyword, the database administrator can explicitly specify the list of segment types to include in the propagation data stream. With a SEGEXCL= keyword, the database administrator can explicitly specify the list of segment types of this physical database to exclude from the propagation data stream.

If it is important for the applications accessing the target DB2 tables that IMS changes of different IMS segment types be applied to DB2 in the same sequence as the original IMS DB updates, include the changed data of these IMS segment types in the same PRSTREAM. For example, the propagation of the entity segment type and of all extension segment types of mapping case 2 should be transmitted as part of the same PRSTREAM.

```
►►─DB──┬─DBD=dbdname──────────────────────────────┬──;──────────────────►◄
       │                  ┌─,──────────┐          │
       │                  │            ▼          │
       │                  ├─,SEG=(──segname──)────┤
       │                  │            ┌─,──────┐  │
       │                  │            ▼        ▼  │
       │                  └─,SEGEXCL=(──segname──)─┘
       │          ┌─,──────┐
       │          ▼        ▼
       ├─DBD=(──dbdname──)──────────────────────────┤
       └─ALL──────────────────────────────────────┘
```

ALL
Specifies that all changes captured by the IMS Capture Exit Facility for the EKYMQCAP Exit Routine be included in the propagation data stream.

DBD= *dbdname*
Identifies the dbdname of a physical database whose captured data is transmitted as part of the preceding PRSTREAM.

SEG= *(segname1,segname2,...)*
Identifies the segment names whose captured data is transmitted as part of the preceding PRSTREAM.

SEGEXCL= *(segname1,segname2,...)*
Identifies the segment names whose captured data is not transmitted as part of the preceding PRSTREAM.

For each IMS database, do not provide more than one DB control statement specifying the IMS database name on the DBD= keyword for the same PRSTREAM.

FIELD control statements

The FIELD control statement is optional. By default, when capturing the data of a segment type, the Capture component transmits the changed data of each Replace database operation to the target system.

You might not want to propagate the changes of all fields of a segment type. In some cases, you might experience a significant performance difference if:
- You transmit all Replaces to the Apply programs, or
- You transmit only Replaces that change the data of fields you have specifically identified.

In the latter case, the FIELD control statements are useful. They identify fields of a segment type that are relevant for propagation. During a Replace, if none of these fields has changed, the Replace is not sent to the Apply program. If at least one of these fields has changed, the Replace is sent.

Note: Inserts and Deletes are always sent to the Apply program.

Recommendation: You should provide FIELD control statements *only* when they *substantially* reduce the amount of data transmitted to the Apply program, because:
- The FLDSTRT= and FLDLEN= specifications might need to be adapted when the IMS segment layout changes. If you forget to adapt these specifications (or if these specifications are not correct), the incorrect subset of data can be transmitted to the Apply program.
- Use of FIELD control statements increases the virtual storage requirements and the amount of CPU processing of IMS DPROP.

FIELD control statement specifications apply to specific propagation data streams. They must be provided after the PRSTREAM control statement to which they apply (and before the next PRSTREAM control statement).

Syntax of FIELD control statement

```
►►──FIELD──DBD=dbdname──,SEG=segname──,FLDSTRT=startbyte──,FLDLEN=field_length──────►
```

```
►──┬──────────────────────┬──;──────────────────────────────────────────►◄
   └─,FLDNAME=field_name──┘
```

DBD= *dbdname*
Identifies the name of the physical IMS database that contains the field.

SEG= *segname*
Identifies the name of the IMS segment that contains the field.

FLDSTRT= *start_byte*
Specifies the start position of the field within the segment. With IMS DPROP, the first byte of a field is considered to have the start position of 1, not 0.

FLDLEN= *field_length*
Specifies the length of the field, expressed in the number of bytes.

FLDNAME= *field_name*
Specifies the name of the field for documentation purposes.

Note: For contiguous fields, you can provide either one FIELD control statement for each field, or, one FIELD control statement for the group of contiguous fields.

KEYRANGE control statement

This control statement is optional. KEYRANGE enables the Capture component to selectively include changed segment occurrences of a database in the PRSTREAM.

Use KEYRANGE to limit the inclusion of changed segments in the PRSTREAM by identifying only those segments of a database that have root-key values within the specified keyrange. With this control statement, you can effectively split large databases, sending them through multiple PRSTREAMs to the target location. Performance is enhanced, because you can run multiple Apply programs at the target location. You can also use KEYRANGE to distribute changed data to different locations based on root-key criteria.

You can provide multiple KEYRANGE statements for each PRSTREAM, each specifying a different databases.

```
►►─KEYRANGE──┬──────────────────────────────┬──┬──────────────┬──┬──────────────┬──►◄
             │                ┌──,──────┐   │  └─,LKEY=lowkey─┘  └─,HKEY=highkey─┘
             └─DBD=(──────────▼─dbdname──┘─)─┘
```

DBD=
Identifies the names of the databases for which this KEYRANGE control statement applies. If a DBD is not named, the KEYRANGE control statement applies to all databases for which captured data is transmitted by this PRSTREAM.

LKEY=
Is a character string of up to 256 bytes that specifies the lowest key to be included in the selection. All segments with a root-key less than this value are not included in the PRSTREAM. If an LKEY is not provided, all database segments up to the highest key (HKEY) value are included in the PRSTREAM.

LKEY can be specified as a hexadecimal value.

HKEY=
Is a character string of up to 256 bytes that specifies the highest key to be

included in the selection. All segments with a root-key higher than this value are not included in the PRSTREAM. If an HKEY is not provided, all segments from the lowest key specified to the end of the database are included in the PRSTREAM.

HKEY can be specified as a hexadecimal value.

Exploitation of VLF by the MQ-ASYNC Capture Exit Routine

The Capture exit routine of MQ-ASYNC exploits the functionality of the OS/390 Virtual Lookaside Facility (VLF) to get immediate notification of:

- An emergency stop (and a restart/reset) of the Capture activities. With the CUT utility, an operator can request an emergency stop and a restart/reset of the Capture activities in all updating IMS jobsteps that access the same EKYMQST DPROP status file.
- A change of the transmission specifications in the EKYTRANS file.

Defining a VLF PDS class in SYS1.PARMLIB for an MQ-ASYNC Capture system is a basic activity for an OS/390 system programmer. This definition is not required, but it is recommended. When the VLF PDS class is defined and active, the Capture exit routine checks frequently and efficiently for changes in the DPROP status file and the EKYTRANS file.

After the VLF PDS class reserved for an MQ-ASYNC Capture system is properly defined in SYS1.PARMLIB and is active, MQ-ASYNC detects changes as follows:

1. The EKYMQCAP exit routine checks the VLF copy of the EKYMQST status file to detect an emergency stop and a reactivation/reset of the Capture activities.
2. the EKYMQCAP exit routine check the VLF copy of the transmission specifications to detect a change of the transmission specifications in the default EKYTRANS PDS of the MQ-ASYNC Capture system.

   In each case, the VLF copy is checked:
   - When invoked the first time within each IMS unit-of-work, and
   - For long running updating IMS units-of-work, approximately every 10 seconds.

If the VLF PDS class reserved for an MQ-ASYNC Capture system has *not* been defined or is not active, MQ-ASYNC detects changes as follows:

1. To detect an emergency stop and a reactivation/reset of Capture activities, the EKYMQCAP exit routine checks approximately every minute if the status file has changed.
2. To detect a change in the transmission specifications, the EKYMQCAP exit routine checks approximately every minute whether the specifications have changed.

When the transmission specifications are not provided in the default EKYTRANS PDS of the MQ-ASYNC Capture system, the EKYMQCAP exit routine cannot exploit VLF to access the transmission specifications. In this case, the EKYMQCAP exit routine also checks every minute for changes in the transmission specifications.

Chapter 15. Apply Program

The Apply program is a batch program that:

- Uses MQSeries to read the MQSeries messages sent by the Capture component and by the event marker.
- Calls the IMS DPROP Relational Update Program (RUP) to transform the changed IMS data into relational format and to update the DB2 target tables.

The Apply program uses both MQSeries and DB2 services. For integrity reasons, the MQSeries and DB2 activities of the Apply program are coordinated with a two-phase commit protocol. The commit coordination is performed either by IMS (if the Apply program executes as an IMS BMP) or by RRS (if the Apply program executes as an IMS batch program or a non-IMS batch program).

The Apply program occurrences are started like other batch jobs or tasks in your installation (often with standard automated operation tools of your installation).

Apply program occurrences are stopped either:

- With an OS/390 operator 'STOP jobname' command, or
- When reading an MQSeries message whose source system timestamp is equal to or greater than a specifiable timestamp.
- When reading an event marker MQSeries message identifying a source system point-in-time, at which time the Apply program stops. This is often preferable to stopping at an explicitly specified timestamp.
- When the MQSeries Queue Manager stops.
- When DB2 stops, if Apply is running in a non-IMS environment. If it is running as an IMS Program, the Apply program stops/abends only if DB2 is stopped with MODE(FORCE).

When used for near-real-time propagation, the Apply program occurrences are permanently active. When used for point-in-time propagation, the Apply program occurrences are started by the system operator when the target tables must be updated. The Apply program occurrences can automatically stop when processing the event marker MQSeries message identifying the source system point-in-time that must be reflected in the content of the target tables.

You can run either one single occurrence or multiple occurrences of the Apply program. Running multiple Apply program occurrences increases throughput. To run multiple Apply occurrences concurrently, consider splitting the transmission of the changed IMS data into multiple PRSTREAMs by using KEYRANGE= control statements.

Note: A particular MQSeries input queue and a particular PRSTREAM cannot be concurrently processed by more than one Apply program occurrence. This preserves the FIFO processing sequence of the MQSeries messages containing the changed data.

An Apply program occurrence uses a control statement file to understand which MQSeries queue is used as input. This MQSeries queue is opened for exclusive use by the executing Apply program occurrence.

You can use the OS/390 operator command 'MODIFY jobname' to get information about:

- How many MQSeries messages the Apply program has processed
- The last processed MQSeries message of each input propagation data stream.

The following topics provide additional information:
- "Processing of MQSeries Messages Containing IMS changes"
- "Processing of MQSeries Messages Containing Event Marker"
- "JCL for the Apply Program"
- "Control statements in the //APPLYIN file" on page 379
- "OS/390 Operator Commands for the Apply Program" on page 390
- "Special DB2 Tables of the Apply Program" on page 392

Processing of MQSeries Messages Containing IMS changes

When the Apply program receives an MQSeries message containing an IMS change, it calls the IMS DPROP RUP to map the IMS data into relational format and update the target tables.

By default, when the //APPLYIN file contains neither INCLUDE nor EXCLUDE control statements, the RUP processes all Propagation Requests (PRs) propagating the changed IMS segment type. Use the INCLUDE and EXCLUDE statements to explicitly specify which PRs the RUP will process.

The EXCLUDE control statement can also be used to request that all PRS, or a subset of PRs, be bypassed or excluded until a specific source system point-in-time. This is useful when synchronizing or re-synchronizing source copies with their target copies, and when migrating from LOG-ASYNC.

Processing of MQSeries Messages Containing Event Marker

When reading an MQSeries message containing an event marker, the Apply program stores the event marker into a special MQ-ASYNC event marker table and issue a WTO to inform the operators of receipt of the event marker.

Depending on specifications of the optional STOPAT control statements in the //APPLYIN file, the Apply either stops or continues to read and process MQSeries messages.

Note that before deciding whether to stop, the Apply program checks for duplicate Apply occurrences (an Apply program with the same Apply name) that have already received and processed an event marker via another propagation data stream. Duplicate event marker messages are not processed.

The Apply program periodically deletes those rows that are older then one month from the MQ-ASYNC event marker DB2 table.

JCL for the Apply Program

The Apply program can run:
- as a non-IMS batch program
- as an IMS BMP program
- as an IMS batch program

If you are executing the Apply program in an IMS environment, use the JCL procedures provided by IMS for IMS BMP jobsteps or IMS batch jobsteps.

If you are executing the Apply program as an IMS Batch jobstep, specify RRS=Y when calling the DLIBATCH or the DBBBATCH JCL procedure.

Following are examples of JCL used for the Apply program when running in a non-IMS environment and when running in an IMS BMP environment:

```
//APPLY    EXEC  PGM=EKYI300X
//STEPLIB  DD    ......
//         DD    ......
//EKYRESLB DD    DSN=PREFIX.EKYRESLB,DISP=SHR
//APPLYIN  DD    *
APPLY NAME=APPLY01 ;
QMANAGER NAME=MQ01 ;
QUEUE NAME=MQ.APPLY.NY.QUEUE01 ;
DB2 SYSTEM=DSN , PLAN=EKYI300X ;
//EKYIN    DD    *
//EKYPRINT DD    SYSOUT=A
//EKYTRACE DD    SYSOUT=A
```

Figure 272. Example of JCL for Apply when running in a non-IMS environment

```
//APPLYBMP     EXEC     IMSBATCH,
//      MBR=EKYI300X,
//      PSB=APPLY01
//*
//STEPLIB   DD    .....
//          DD    .....
//EKYRESLB  DD    DSN=PREFIX.EKYRESLB.EKYRESLB,DISP=SHR
//DFSESL    DD    .....
//          DD    .....
//APPLYIN   DD    *
APPLY NAME=APPLY01 ;
QMANAGER NAME=MQ01 ;
QUEUE NAME=MQ.APPLY.NY.QUEUE01 ;
//EKYIN     DD    *
//EKYPRINT  DD    SYSOUT=*
//EKYTRACE  DD    SYSOUT=*
```

Figure 273. Example of JCL for Apply when running as an IMS BMP

//STEPLIB If the IMS DPROP modules, your IMS DPROP exit routines, DB2 modules, or MQSeries modules are not in the link pack area or in a link-list library, you must provide a //STEPLIB or //JOBLIB to identify the library in which they are located.

//EKYRESLB This DD statement is common to many IMS DPROP components. See “//EKYRESLB DD Statement” on page 4 for more information.

//DFSESL This DD statement might be required by IMS if you run the Apply program as an IMS BMP. See “//DFSESL DD Statement ” on page 4 for more information.

//APPLYIN This defines the mandatory file where you provide control statements for the Apply program.

//EKYIN This defines an optional file for providing TRACE and TRDEST control statements. This is used to activate a trace for problem determination purposes.

//EKYPRINT This defines a sysout data set containing messages from the Apply program and from other IMS DPROP components.

//EKYTRACE This defines the usual trace data set of IMS DPROP. See "//EKYTRACE DD Statement " on page 7 for more information.

//APPLYIN DD Statement

This statement is mandatory. The //APPLYIN file contains the most important definitions of the MQ-ASYNC Apply program. "Control statements in the //APPLYIN file" on page 379 provides more information on the control statements of the //APPLYIN file.

Each occurrence of the Apply program must use its own //APPLYIN file.

//APPLYIN must be a sequential file or a PDS member and must have the following characteristics: RECFM=F or FB, and LRECL=80.

//APPLYIN contains the following optional control statements to determine what to process and when to stop:
- 1-*n* INCLUDE control statements
- 1-*n* EXCLUDE control statements
- 1-*n* STOPAT control statements

//APPLYIN contains the following mandatory control statements:
- 1 APPLY control statement
- 1 QMANAGER control statement
- 1 QUEUE control statement
- 1 DB2 control statement (only if the Apply program is not running as an IMS jobstep)

//APPLYIN contains the following additional optional control statements:
- 1 COMMIT control statement
- 1 STATISTIC control statement

The following are examples of an //APPLYIN DD statement:

```
//APPLYIN   DD   DSN=IMSDPROP.APPLYDEF(APPLY01),DISP=SHR
```

Figure 274. First example of an //APPLYIN DD Statement

```
//APPLYIN   DD   *
APPLY NAME=APPLY01 ;
QMANAGER NAME=MQ03 ;
QUEUE NAME=MQ.APPLY.NY.QUEUE01 ;
DB2 SYSTEM=DSN,PLAN=EKYAPPLY;
/*
```

Figure 275. Second example of an //APPLYIN DD Statement

//EKYIN DD Statement

The optional //EKYIN statement provides IMS DPROP tracing-related specifications.

For the Apply component of MQ-ASYNC, the following control statements can be provided in the //EKYIN file:

- TRACE
- TRDEST

Other JCL Statements for APPLY Component of MQ-ASYNC

Other JCL statements for the Apply program include:

- //EKYRESLB
- //EKYPRINT
- //EKYLEOPT
- //JOBLIB or //STEPLIB
- //EXEC

Control statements in the //APPLYIN file

APPLY control statement

The APPLY control statement is mandatory. You must provide one, and only one, APPLY control statement in the //APPLYIN file.

The APPLY control statement assigns a unique name to each occurrence of the Apply program. When starting, the APPLY program verifies that another occurrence of the Apply program with the same name is not currently executing.

```
►►──APPLY_NAME=name_of_apply_program──;──────────────────────────────────────►◄
```

NAME= *name_of_apply_program*
: Assigns a unique name to the Apply program occurrence. The name must consist of 1-8 alphanumeric or the $, #, @ special characters. The first character cannot be numeric.

COMMIT control statements

The optional COMMIT control statement overrides the default commit frequency of the Apply program. If you provide multiple COMMIT control statements, specifications on the *n*th control statement override specifications on the *m*th control statement (n>m).

```
►►──COMMIT──AFTER=nnnnn──;──────────────────────────────────────────────────►◄
```

AFTER= *nnnnn*
: Specifies that the Apply program commits, approximately, after processing *nnnnn* database updates.

  The default value is 500. The maximal value is 999999.

Note: These values are not the exact number of database updates when the Apply program issues a commit. Because the Apply program always processes entire MQSeries messages for commits, the commit point can vary and be slightly higher than the selected value.

To avoid enqueueing DB2 resources for long periods, the Apply program also goes through commit processing after each couple of seconds, even if it has not yet processed the number of database updates.

DB2 control statement

If you are running the Apply program in a non-IMS region, you must provide one, and only one, DB2 control statement in the //APPLYIN file. For IMS Regions, do not provide a DB2 control statement.

The DB2 control statement identifies which DB2 system and which DB2 plan are used for the execution of the Apply program.

Syntax of DB2 control statement

```
►►──DB2──SYSTEM=db2_subsystem──,PLAN=db2_plan──;──────────────────────────►◄
```

SYSTEM= *db2_subsystem*
Identifies the name of the DB2 subsystem (or name of a DB2 group attachment, if used in a DB2 data sharing group) for use by the Apply program.

PLAN= *db2_plan*
Identifies the name of the DB2 plan for use by the Apply program.

The DB2 plan must provide access to the package collections containing the following packages:

- packages of the SQL updates modules generated as part of the PR definitions
- packages of IMS DPROP exit routines
- packages for RUP's access to the IMS DPROP directory tables
- packages for the Apply program's access to DB2 tables

EXCLUDE control statements

This statement is optional. By default, if you provide neither EXCLUDE nor INCLUDE, the APPLY Program processes all PRs that propagate the changed IMS segments contained in the MQSeries messages.

The database administrator can override this default by providing one or multiple EXCLUDE control statements that identify explicitly which PRs should *not* be processed. The EXCLUDE control statement can identify these PRs through the use of the PR=, PRSET=, DBD=, and SEG= keywords.

When combining INCLUDE and EXCLUDE control statements, specifications of EXCLUDE override specifications of the INCLUDE control statement.

The following is a diagram of the EXCLUDE control statement.

```
►►──EXCLUDE──┬──────────────────┬──────────────────────────────────────────►
             └─UNTIL=timestamp,─┘
```

```
►──┬─,PR=(──┬─pr_id──┬─)──────────────────────────────────────────────────────────────;──►◄
   │        └──,──────┘                                                                  │
   ├─,PRSET=(──┬─prset_id──┬─)──────────────────────────────────────────────────────────┤
   │           └──,────────┘                                                             │
   ├─,DBD=(──┬─dbd_name──┬─)──┬──────────────────────────────────────────────────────────┤
   │         └──,────────┘    │                                                         │
   │                          └─,PRSET=(──┬─prset_id──┬─)──┘                            │
   │                                      └──,────────┘                                  │
   ├─,DBD=dbd_name,SEG=(──┬─seg_name──┬─)──┬──────────────────────────────────────────┤
   │                      └──,────────┘    │                                           │
   │                                       └─,PRSET=(──┬─prset_id──┬─)──┘              │
   │                                                   └──,────────┘                    │
   └─,ALL──────────────────────────────────────────────────────────────────────────────┘
```

UNTIL= *timestamp*

Tells the Apply program to exclude the processing of the identified PRs for all MQSeries messages that have been created before the specified source system timestamp.

The source system timestamp is specified in terms of local time of the source system and must be provided in the following format: yyyy-mm-dd-hh.mm.ss.nnnnnn. The year, month, and day must be specified. If your timestamp omits trailing hour, minute, second, or microsecond specifications, implicit specifications of zeroes are assumed for the missing parts.

On the Control statement, you must provide a PR Identification Block. The PR Identification Block is described below.

The Propagation Request Identification Block

The combination of the PR, PRSET, DBD, SEG, and ALL keywords comprise a *PR identification block*. Use the propagation request identification block to describe to which propagation requests the control statement applies.

PR=(*prid1,prid2,...* **)**

Names a propagation request or a list of propagation requests.

PRSET=(*prset1,prset2,...* **)**

Names a propagation request set or a list of propagation request sets. By using this keyword alone or together with a DBD keyword, you can limit the scope of the control statement to the propagation requests that belong to the specified PRSET.

DBD=(*database1,database2,...* **)**

Names a physical IMS database, or a list of physical IMS databases.

When you use this keyword, the control statement applies to all propagation requests propagating the specified IMS databases.

DBD= *database* **, SEG=(***segname1,segname2,...***), PRSET=(***prset1,prset2,...***)**

Names a physical IMS segment type, or a list of IMS segment types.

If you use the SEG keyword, you must specify one physical IMS database name on the DBD keyword. When you specify the combination of DBD= and SEG= keywords, the control statement applies to all propagation requests propagating the specified IMS segment types.

Do not specify the name of internal segments in the SEG= keyword. Internal segments are not IMS segments; they represent structures that are embedded within a containing IMS segment, that are propagated by mapping case 3 propagation requests.

When providing a control statement for propagation requests of internal segments, specify either:
- The propagation request IDs on the PR= keyword
- The name of the containing IMS segment in the SEG= keyword

ALL
Specifies all propagation requests of an IMS DPROP system.

Internal Segments

If you specify the name of an IMS segment that contains internal segments in the SEG= keyword, the control statement is applied to:
- All propagation requests for the internal segments
- All propagation requests for the containing IMS segment

Path Data

If an IMS segment specified in the SEG= keyword contains fields used as path data, the control statement is:
- applied *only* to those propagation requests that propagate the identified segment as an entity or extension segment, and to those propagation requests that propagate internal segments contained in the identified IMS segment.
- *Not* applied to those propagation requests that propagate dependent segments and include path data from the identified segment.

FAILURES control statements

The optional FAILURES control statements are used to override default options or settings of the error handling logic of the Apply program. They can be used to:
- Tell the APPLY program to accept a limited number of errors.
- Tell the Apply program to limit the number of propagation failures that will be documented with error messages, inserts into the ERROR Table, and traces.

Error Categories Recognized by Apply:

The Apply program distinguishes between the following category of errors:
- Mapping Errors
- Unavailability problems
- Deadlocks and Timeouts
- Miscellaneous Errors
- Should-Not-Occur Errors

Mapping Errors: Mapping errors include situations where the changed source data cannot be mapped or propagated to the target DBMS. Examples of mapping errors include:
- An IMS field that is supposed to contain numeric data but contains non-numeric data and is mapped to a numerical DB2 column. Because it contains non-numeric data, it cannot be mapped to a numerical DB2 column.
- The Apply program is processing the Replace of an IMS segment occurrence and attempts to replace the corresponding DB2 row; however, the corresponding DB2 row does not exist.

Default Error Options:

If no FAILURES control statements are provided, mapping errors result in the writing of error messages and traces. Then, the Apply program will abend, unless the Propagation Request (PR) encountering the error has been defined with ERROPT=IGNORE.

For PRs defined with ERROPT=IGNORE, the Apply program does not abend. Instead, it writes error messages and traces and then inserts the database change that could not be propagated into an ERROR Table. Apply continues its processing with the next IMS database change (or with the next PR, when an IMS database change is propagated by more then one PR).

Note: The ERROPT=IGNORE option is specified at the PR level when the PR is defined. ERROPT=IGNORE is useful for individual PRs, when new PR definitions have not yet been thoroughly tested or when it is known that due to some corrupted data, particular PRs will sporadically encounter problems that should not affect the processing of other PRs.

Overriding the Default Error options:

The defaults actions can be overridden by providing a FAILURES CATEGORY=MAPPING, ACCEPT=*nnnn*; control statement. This control statement tells the Apply program to accept (for PRs that are not defined with ERROPT=IGNORE) the specified number of mapping errors within a 15-minute interval. After writing error messages and traces, the Apply program will test the number of mapping errors encountered (by PRs that are not defined with ERROPT=IGNORE) within the last 15-minute cycle. If the limit specified on the ACCEPT= keyword has not been exceeded, the Apply program writes the change that could not be propagated to an ERROR Table and continues its processing with the next database change (or with the next PR, when an IMS database change is propagated by more than one PR). Otherwise, if the limit specified on the ACCEPT= keyword has been exceeded, the Apply program abends.

FAILURES CATEGORY=MAPPING, ACCEPT=*nnnn*; is an error option at the Apply program level and is useful in emergency situations, when encountering unexpected errors. Its use continued processing when there are a limited number of unexpected errors. These errors might not necessarily be associated with a particular PR. For example, after a cold start of MQSeries or of a MQSeries link, the Apply program might attempt to replace a row that has never been inserted because the insert got lost as part of the cold start.

Unavailability Problems: Examples of unavailability problems include the unavailability of a particular DB2 table or DB2 table space. There are also other errors that DPROP treats as unavailability problems: for example, a mismatch of the IMS DBD Version ID when the PR was defined and the IMS DBD Version ID during the database change. To avoid these problems, the operator or database administrator can make a PR defined with the new DBD Version ID available to the Apply program. *Default Error options*:

when the Apply program encounters an unavailability problem, it writes error messages and abends. The operator should investigate the reported problem, make the resource available, and restart the Apply program.

Overriding the Default Error Options:

The defaults action can be overridden by providing a FAILURES CATEGORY=UNAVAILABLE, ACCEPT=NNNN control statement. The Apply program processes the control statement in the same way as for mapping errors, described earlier in this topic.

Miscellaneous Errors: Examples of what the Apply calls miscellaneous errors include the reading of an MQSeries message that has not been created by IMS DPROP (this occurs with MQSeries queues that are not used exclusively by IMS DPROP). It also includes readings of MQSeries messages that do not have the expected IMS DPROP message format. *Default Error options*:

When the Apply program encounters a miscellaneous problem, it writes error messages and abends. The operator should investigate the reported problem, fix the problem, and restart the Apply program.

Overriding the default error options:

The defaults actions can be overridden by providing a FAILURES CATEGORY=MISC, ACCEPT=*nnnn*; control statement. The Apply program processes the control statement in the same way as for mapping and unavailability errors, described earlier in this topic.

Deadlocks and Timeouts: For DB2 deadlock and timeout errors, the Apply program will retry the processing of the unit-of-work that deadlocked or timed-out, without abending.

Should-Not-Occur Errors: A should-not-occur error involves an IMS DPROP logic error. When the Apply program encounters this type of error, it writes error messages and abends.

Unlike other errors, you cannot use the FAILURES control statement to avoid abends for this type of error.

Warnings:

1. Use of FAILURES CATEGORY=*xxxxxx*, ACCEPT=*nnnnn*; control statements typically results in IMS changes that are not propagated (instead, these IMS Changes are stored for problem determination purposes by the Apply program in a special DB2 table: the MQ-ASYNC ERROR Table). Because some changes are not propagated, the source and target copy of the propagated data become inconsistent. To reinstate the consistency, you must use the CCU or repeat an extract/load process, which can be time consuming.

   Recommendation: Use the FAILURES CATEGORY=*xxxxxx*, ACCEPT=*nnnnn*; control statements only as a last resort or when the consistency of source and target copies is not important.

   If you are propagating data for production use, use the FAILURES CATEGORY=*xxxxxx*, ACCEPT=*nnnnn* control statements only when encountering problems that cannot be resolved or bypassed by other methods.
2. With FAILURES CATEGORY=*xxxxxx*, ACCEPT=*nnnnn*; control statements, and when defining PRs with ERROPT=IGNORE, the Apply program, by default, documents every encountered error, up to a maximum of 100,000 errors per 15-minute error reporting cycle. When *nnnnn* is a large number, or when PRs defined with ERROPT=IGNORE encounter a large number of mapping errors, this can result in a considerable number of error messages, traces, and inserts into the ERROR Table.

   You can limit the number of error messages, traces, and inserts written to the ERROR Table by using the MSGWTO=*nnn*, MSGPRINT=*nnn*, MSGAUD=*nnn*, TRACE=*nnn*, and INSERT=*nnn* keywords of the FAILURES control statement.

Note: When the Apply program abends because of a mapping, unavailability, or miscellaneous error, its abend message identifies the category of the encountered error. The reported error category can then be used to specify

the CATEGORY= keyword value on a FAILURE control statement, if the operator decides to restart the Apply program with a FAILURE control statement.

If you provide multiple FAILURES control statements, specifications on the *n*th control statement override specifications on the *m*th control statement (n>m).

Syntax of FAILURES control statement

```
►►─FAILURES──┬─CATEGORY=DATA, ACCEPT=9999──────────┬──┬─────────────────────────►
             ├─CATEGORY=UNAVAILABLE, ACCEPT=9999──┤  └─,MSGWTO=9999──┘
             └─CATEGORY=MISC, ACCEPT=9999─────────┘

►──┬──────────────┬──┬──────────────┬──┬──────────────┬──┬──────────────┬──;──►◄
   └─,MSGPRINT=9999─┘  └─,MSGAUD=9999─┘  └─,TRACE=9999─┘  └─,INSERT=9999─┘
```

CATEGORY= *xxxxx* **, ACCEPT=***9999*

Tells the Apply program to accept, within each 15-minute cycle, up to the specified number of errors of the error category *xxxxx*.

Values for *xxxxx* include MAPPING, UNAVAILABLE, or MISC.

Substitute 9999 by a number in the range between 0 and 999999999. The default value is 0.

If you ever need to accept errors of more then one category, which is unlikely, you can provide FAILURE control statements for each error category.

The keyword specifications that follow affect the error handling of all errors and error categories, not only the error category identified on the optional CATEGORY= keyword.

MSGWTO= *9999*

Tells the Apply program to report within each 15 minute cycle, up to the specified number of errors with WTOs (write-to-operators) on the /390 operator consoles.

The maximum number that can be specified is 999999999. The default value is 50.

Note: IMS DPROP also reports errors on a print file, on its trace file, and on its audit trail. If encountering more errors than the MSGWTO specification, the Apply still reports errors on its print file, trace file, and audit trail (assuming that the MSGPRINT=, TRACE=, and MSGAUD= limits have not been exceeded).

MSGPRINT= *9999*

Tells the Apply program to report within each 15 minute cycle, up to the specified number of errors on the //EKYPRINT Print file.

The maximum number that can be specified is 999999999. The default value is 100000.

MSGAUD= *9999*

Tells the Apply program to report within each 15 minute cycle, up to the specified number of errors (via SMF records) for the IMS DPROP Audit Trail.

The maximum number that can be specified is 999999999. The default value is 100000.

TRACE= *9999*

Tells the Apply program to document within each 15 minute cycle, up to the specified number of errors on the IMS DPROP trace.

The maximum number that can be specified is 999999999. The default value is 100000.

INSERT= *9999*

Tells the Apply program to insert, within each 15 minute cycle, up to the specified number of IMS changes that could not be propagated into the ERROR Table.

The maximum number that can be specified is 999999999. The default value is 100000.

INCLUDE control statements

This control statement is optional. By default, when you provide neither INCLUDE nor EXCLUDE control statements, an APPLY program occurrence processes all PRs that propagate the changed IMS segments contained in the MQSeries messages that it reads.

The database administrator can override this default by providing one or multiple INCLUDE control statements that identify explicitly which PRs shall be included in the Apply processing. The INCLUDE control statement can identify these PRs through use of PR=, PRSET=, DBD= and SEG= keywords.

When combining INCLUDE and EXCLUDE control statements, specifications of the EXCLUDE control statements will override specifications of the INCLUDE control statements.

The syntax of the INCLUDE control statement is shown in the following diagram.

```
                    ┌─,──────────┐
►►─INCLUDE─,DBD=(──┴─dbd_name──┴─)──┬──────────────────────────────────────┬──;──►◄
                                    │                                      │
                                    └─,DBD=dbd_name,SEG=(──┬─,──────────┬──┴─)──┘
                                                           └─seg_name───┘
```

On the control statement, the word INCLUDE is followed by a Propagation Request (PR) Identification Block. The PR Identification Block, and related information on Internal Segments and Path Data, are described in “The Propagation Request Identification Block” on page 381.

QMANAGER control statement

This statement is mandatory. You must provide one, and only one, QMANAGER control statement in the //APPLYIN file.

The QMANAGER control statement identifies the MQSeries queue manager used to read the MQSeries messages containing the changed data sent by the Capture component.

Syntax of QMANAGER control statement

```
►►──QMANAGER_NAME=name_of_mqseries_queue_manager──;──────────────────────────►◄
```

NAME= *name_of_mqseries_queue_manager*
Identifies the name of the MQSeries queue manager that will be used by the Apply Component of MQ-ASYNC to read the changed data sent by the Capture component.

QUEUE control statement

The mandatory QUEUE control statement identifies which MQSeries input queue contains the propagation data streams that must be processed by the Apply program.

Note that the queues used by MQ-ASYNC must be reserved for exclusive use by MQ-ASYNC. Also, in order to preserve the FIFO message processing sequence, the Apply program opens its MQSeries input queue with the INPUT_EXCLUSIVE option of MQSeries. Therefore, this queue cannot be processed concurrently by more then one occurrence of the Apply program.

Each occurrence of the Apply program can read from one and only one MQSeries input queue. One MQSeries input queue can contain one or multiple propagation data streams.

Note, however, that when an MQSeries input queue contains multiple propagation data streams that have not been transmitted from source to target via the same MQSeries queues, use of the STOPAT control statement might not result in a stop of the Apply at the desired source system point-in-time.

Syntax of QUEUE control statement

```
►►──QUEUE_NAME=name_of_mqseries_input_queue──;──────────────────────────────►◄
```

NAME= *name_of_mqseries_input_queue*
Identifies the name of the MQSeries queue (or of an MQSeries QALIAS) that will be used as input by the Apply program.

STOPAT control statements

The optional STOPAT control statement is used to automatically stop the Apply program:

- When it reads an MQSeries message with a source system timestamp higher than the specified source system timestamp, or
- When it reads an MQSeries message containing a source system event marker having a specified event marker identification.

After this automated stop of the Apply program, the data in the DB2 target tables reflects the content of the source databases at the specified source system timestamp or when the event marker was created on the source system. The STOPAT control statement can therefore be used to implement propagation that reflects, in the content of the target DB2 tables, a clear source system point-in-time.

If you provide multiple STOPAT control statements in the //APPLYIN file, the Apply program stops when encountering the first MQSeries message that satisfies at least one of the STOPAT control statement specifications.

Guidelines for Using STOPAT

1. To reflect in the target table an accurate source system point-in-time, the source IMS databases should be quiesced at the specified point-in-time (at this point-in-time, there should be no active UOW that has updated or will update the source databases).

   If this guideline is not observed and some updating UOWs are active at the specified point-in-time, some of the updates of these UOWs might not be on the target tables once the Apply program stopped. However, they are not lost; they remain in the MQSeries queue until the Apply program is started again and processes them at this later time.

   In addition, if this guideline is not followed, some IMS UOWs might start or complete either before or after the specified point-in-time; some other IMS UOWs might overspan the point-in-time. Therefore, updates of a particular UOW might or might not be included in the target tables when the Apply program stops.
2. Do not use the STOPAT control statement for an Apply program that processes multiple propagation data streams that have not been transmitted together in FIFO order from the same source system queue.

   If this guideline is not observed, the following can happen when the Apply program is told to stop at 10:00: the Apply program might read a message from stream 1 written at 10:01 before it reads from a message from stream 2 written at 9:58. It then stops when encountering the message written at 10:01 before reading the message written at 9:58.

Notes:

1. The operators of the target system can also stop the Apply Program with an OS/390 Console `STOP jobname` command.
2. You might want to define and leave in the //APPLYIN a STOPAT control statement specifying an agreed-upon event marker identification. This allows the source system operators to stop the Apply at any source system point-in-time, without requiring changes in the //APPLYIN control statements of the Apply program.

```
►►──STOPAT──┬─ID=Id_of_Event_Marker─┬──;──────────────────────────────────►◄
            └─TS=timestamp──────────┘
```

ID= *Id_of_Event_Marker*
Tells the Apply program to stop its processing when reading an MQSeries message containing an event marker with the specified identification.

TS= *timestamp*
Tells the Apply program to stop its processing when reading MQSeries messages that have been created after the specified source system timestamp.

The source system timestamp is specified in terms of local time of the source system and must be provided in the following format: yyyy-mm-dd-hh.mm.ss.nnnnnn. The year, month, and day portions must be specified. If your timestamp omits trailing hour, minute, second or microsecond specifications, implicit specifications of zeroes are assumed for the missing trailing portion.

The logic of the Apply is as follows: When reading an MQSeries message that has been written after the specified timestamp, the Apply program stops its processing without processing this MQSeries message. This message remains in the MQSeries queue and is processed by an eventual execution of an Apply program.

When reading an MQSeries message that has been written exactly at the specified timestamp, the Apply program processes this message and attempts to read subsequent messages. If the next MQSeries message has the same timestamp, it is processed and the Apply program attempts to read subsequent messages. If the next message has a higher timestamp, or if the Apply program does not find any further message to process, the Apply program then stops.

Notes:

1. On the source system, if there was a long period of no capture activities at and after the specified timestamp, the Apply program, when executing in near-real-time mode, waits a long time before it gets an MQSeries message that has been written on the source system after the specified timestamp. If it is important to immediately stop the Apply program, you can use the `STOP Jobname` command.

   Use of the STOPAT EM= control statement does not have this disadvantage.

STATISTIC control statements

The optional STATISTIC control statement is used to override the default options for the periodic writing of performance statistics.

If you provide multiple STATISTIC control statements, specifications on the *n*th control statement override specifications on the *m*th control statement (*n*>*m*).

Syntax of STATISTIC control statement

```
►►─STATISTIC─┬──────────────────────────┬──────────────────────────────────────►
             └─FREQUENCY=nnnnnn─────────┘

►─┬──────────────────────────────────────────────────────────────────────────┬─►◄
  └─,DETAIL=─(─┬─PSB─┬─,─┬─────┬─,─┬─────┬─)─┘
               ├─DBD─┤   ├─PSB─┤   ├─PSB─┤
               └─SEG─┘   ├─DBD─┤   ├─DBD─┤
                         └─SEG─┘   └─SEG─┘
```

FREQUENCY= *nnnnnn*

Specifies that it is after each period of nnnnnn seconds that the Apply program will write performance statistic records to SMF.

The default value is 3600 (i.e. each hour). The maximum value is 999999.

Note: The Apply program writes final statistics at the time of its normal termination.

DETAIL=

Specifies how detailed the performance statistics should be. DETAIL is any combination, in any order, of the following keywords:

PSB Provides detail at the PSB level

DBD Provides detail at the DBD level

SEG Provides detail at the SEG level

These selections are used to build the key for one report item on the statistics.

If the DETAIL keyword is omitted, then statistic report items are done only per source IMS system ID.

Figure 276 displays the statistics if PSB,DBD,SEG is specified in the control statement.

```
EKYI430I SYS  PSBNAME  DBDNAME  SEGNAME  --------- NUMBER OF DATABASE UPDATES ---------- --------- DELAY ----------
EKYI430I ---- -------- -------- --------      INSERT     REPLACE      DELETE      ERRORS      AVG      MIN      MAX
EKYI430I SY4A PSBT4RPL TPDB004  SEGA              1           1           1           0     2.57     1.90     3.93
EKYI430I SY4A PSBT4RPL TPDB004  SEGB              0           2           0           0     0.98     0.97     0.99
EKYI430I SY4A PSBT4RPL TPDB004  SEGD              2           1           2           0     0.88     0.85     0.92
EKYI430I SY4A PSBT5RPL TPDB005  SEGA              1           1           1           0     2.00     1.40     2.31
EKYI430I SY4A PSBT5RPL TPDB005  SEGB              1           3           1           0     1.96     1.53     2.48
EKYI430I SY4A PSBT5RPL TPDB005  SEGC              1           1           1           0     2.64     2.29     3.34
EKYI430I SY4A PSBT5RPL TPDB005  SEGD              2           1           2           0     2.11     1.79     2.46
EKYI430I SY4A PSBT6RPL TPDB006  SEGA              1           1           1           0     1.57     1.57     1.57
EKYI430I SY4A PSBT6RPL TPDB006  SEGB              1           3           1           0     1.31     0.81     1.67
EKYI430I SY4A PSBT6RPL TPDB006  SEGC              1           1           1           0     1.91     1.56     2.61
EKYI430I SY4A PSBT6RPL TPDB006  SEGD              2           1           2           0     1.54     0.80     1.77
```

Figure 276. Sample Statistics 1

Figure 277 displays the statistics if only DBD is specified:

```
EKYI432I DPROP MQAPPLY <APPLY1  > SUMMARY STATISTIC
EKYI430I SYS  DBDNAME  --------- NUMBER OF DATABASE UPDATES ---------- --------- DELAY ----------
EKYI430I ---- --------      INSERT     REPLACE      DELETE      ERRORS      AVG      MIN      MAX
EKYI430I SY4A TPDB004           11          15          13           3 51461.00     1.29 85591.48
EKYI430I SY4A TPDB005            5           6           5           0     0.99     0.81     1.86
EKYI430I SY4A TPDB006            5           6           5           0     1.94     1.27     3.04
```

Figure 277. Sample Statistics 2

OS/390 Operator Commands for the Apply Program

The Apply program supports the following OS/390 system commands:

- STOP *jobname*
- MODIFY *jobname*,...

In these OS/390 system commands, replace *jobname* by the name of the OS/390 job of the Apply program.

STOP Jobname

The `P jobname` (STOP) OS/390 system command is used to stop the Apply program.

MODIFY Jobname

The `F jobname` (MODIFY) command can be used to:

- Display the current status of the Apply Program
- Display some statistic upon console
- Force a new statistic cycle
- Change the frequency interval of statistics

Syntax of MODIFY Jobname command

```
►►──MODIFY──jobname,──┬─D──┬──────────────────────────────────┬──────────────────────────►◄
                      │    ├─,D=x──────────────────────────────┤
                      │    └─,D=──┬─P─┬──┬──────────────────┬──┘
                      │           ├─D─┤  │                  │
                      │           └─S─┘  └─,T=──┬─T─┬──┬─C─┬──┘
                      │                         │   │  │   │
                      │                         └─L─┘  └─L─┘
                      └─N──┬──────────────────────────────────┘
                           └─,F=nnnnn─┘
```

To display a simple statistic, use `MODIFY jobname,D`. This displays, for each source IMS system ID:

- the number of database changes processed
- the minimum, maximum and average delay in 1/100 of seconds. The propagation delay is the elapsed time between when the message was inserted to the MQSeries Queue on the source system and the time of the database update by the Apply program.

To display a summary statistic that indicates the number MQ messages read and the number of database updates done, use the `MODIFY jobname,D,D=X` command.

For a more detailed statistic, use the `MODIFY jobname,D,D={P|D|S}` command.

D= specifies how detailed the statistics should be:

P Provides detail at the PSB level
D Provides detail at the DBD level
S Provides detail at the segment level

The default statistic is from the start of the Apply program forward (default is T=T). To display only the data gathered since the last statistic cycle has initiated, append the T=C optional keyword to this command. To display the timestamp of the last processed message and its delay, append the T=L optional keyword to this command.

To force the start of a new statistic cycle for the Apply program, use `MODIFY jobname,N`. This writes the current statistics and initiates a new cycle. Additionally, you can change the frequency of the statistic cycle by adding the keyword F=*nnnnn* (where nnnnn is the time of a cycle in seconds) to the command.

Figure 278 on page 392 shows several examples of the Modify command, where modify=F.

```
F jobname,D

EKYI432I DPROP MQAPPLY <APPLY1  > SUMMARY STATISTIC
EKYI430I SYS      UPDATES --------- DELAY ----------
EKYI430I ---- -----------      AVG      MIN      MAX
EKYI430I SY4A          25     1.60     0.62     3.99
EKYI379I NUMBER OF MQGET     :          41
EKYI379I NUMBER OF DB UPDATES:          41

-----------------------------------------------------

F jobname,D,D=S

EKYI432I DPROP MQAPPLY <APPLY1  > SUMMARY STATISTIC
EKYI430I SYS  SEGNAME      UPDATES --------- DELAY ----------
EKYI430I ---- -------- -----------      AVG      MIN      MAX
EKYI430I SY4A SEGA               9     1.67     1.44     1.99
EKYI430I SY4A SEGB              12     1.42     0.66     2.07
EKYI430I SY4A SEGC               5     2.43     1.38     3.99
EKYI430I SY4A SEGD              15     1.41     0.62     2.24

-------------------------------------------------------------

F jobname,D,D=S,T=L

EKYI434I DPROP MQAPPLY <APPLY1  > LAST MESSAGE STATISTIC
EKYI430I SYS  SEGNAME  - LAST MESSAGE TIMESTAMP - --DELAY
EKYI430I SY4A SEGA     2001-09-09-16.38.40.162816     1.44
EKYI430I SY4A SEGB     2001-09-09-16.38.41.211392     0.66
EKYI430I SY4A SEGC     2001-09-09-16.38.40.162816     1.43
EKYI430I SY4A SEGD     2001-09-09-16.38.40.162816     1.65

---------------------------------------------------------

F jobname,D,D=D,T=L

EKYI434I DPROP MQAPPLY <APPLY1  > LAST MESSAGE STATISTIC
EKYI430I SYS  DBDNAME  - LAST MESSAGE TIMESTAMP - --DELAY
EKYI430I SY4A TPDB004  2001-09-09-16.37.59.268352     0.69
EKYI430I SY4A TPDB005  2001-09-09-16.38.41.211392     0.66
EKYI430I SY4A TPDB006  2001-09-09-16.38.12.899840     1.22
```

Figure 278. Modify command examples

Special DB2 Tables of the Apply Program

Notice:

This topic contains general-use programming interface and associated guidance information.

Apply ERRORTAB Table

This topic describes the Apply ERRORTAB Table. This table is used by the Apply program to store MQSeries messages and IMS database changes that it could not process successfully.

Multiple occurrences of the Apply program can share the same ERRORTAB table; or they can have their own distinct ERRORTAB tables. The DB2 plan and/or the

DB2 packages of the Apply program determine which ERRORTAB table an Apply program occurrence will use.

Table 35. Columns of the Apply ERRORTAB Table and Their Data Types

Column	Data Type	Description
DPRNAME	char(8)	IMS DPROP system name of the Apply program
DPRTOKEN	dec(15,0)	IMS DPROP system token of the Apply program
APPLY_TS	timestamp char(26)	timestamp describing when the Apply program encountered the error
APPLY_NAME	char(8)	name of the Apply program
APPLY_JOB	char(8)	jobname of the Apply program
CAPT_MSGTS	timestamp char(26)	timestamp describing when the Capture Component created the MQSeries message
CAPT_CHANGETS	timestamp char(26)	timestamp describing when the IMS change occurred
SYSID	char(8)	System ID describing on which 390 System the MQSeries message was created
IMSID	char(8)	IMS ID describing on which IMS System the MQSeries message was created
DBD	char(8)	IMS DBD name of the changed IMS data
SEG	char(8)	IMS segment name of the changed IMS data
CHANGE_TYPE	char(1)	Type of IMS change: I, D, R, or N (for Insert, Delete, Replace, or No-Op)
PR	char(8)	ID of the Propagation Request (PR) that encountered the error
PSB	char(8)	IMS PSB name of the program updating the source IMS data
PGM	char(8)	name of the program updating the source IMS data
JOB	char(8)	name of the job updating the source IMS data
FCKEY	varchar(200)	Fully concatenated IMS Key of the changed IMS segment
CHANGED_DATA	varchar(1000)	the 1000 first bytes of the changed IMS segment
MSG	varchar(2000)	the 2000 first bytes of the MQSeries message

Note that some columns can have a NULL value. For example, this can happen when the Apply program reads an MQSeries message that has not been created by IMS DPROP.

EVENTMARKER Table

This topic describes the EVENTMARKER Table. This table is used by the Apply program to store the event markers that have been sent via MQSeries messages from the source system.

Multiple occurrences of the Apply program can share the same EVENTMARKER table; or they can have their own distinct EVENTMARKER tables. The DB2 plan and/or the DB2 packages of the Apply program determines which EVENTMARKER table an Apply program occurrence will use.

Table 36. Columns of the EVENTMARKER Table and Their Data Types

Column	Data Type	Description
DPRNAME	char(8)	IMS DPROP system name of the Apply program
DPRTOKEN	dec(15,0)	IMS DPROP system token of the Apply program
APPLY_TS	timestamp char(26)	timestamp describing when the Apply program processed the event marker
APPLY_NAME	char(8)	name of the Apply program
APPLY_JOB	char(8)	jobname of the Apply program
CAPT_MSGTS	timestamp char(26)	timestamp describing when the source system created the MQSeries message containing the event marker
SYSID	char(8)	System ID describing on which /390 System the MQSeries message was created
JOB	char(8)	name of the job that created the event marker
PRSTREAM	char(8)	name of the Propagation Data Stream containing the event marker
EM_TS	timestamp char(26)	timestamp identifying when the event marker was created on the source system
EM_STCK	char(8)	OS/390 Store Clock Value identifying when the event marker was created on the source system
EM_TYPE	char(1)	Type of event marker
EM_ID	char(8)	ID of the event marker

Chapter 16. Capture System Utility (CUT) and Event Markers

The Capture system utility (CUT) is used on the IMS Source system:
- To create event markers that identify a Source system point-in-time to the Apply program.
- INIT DPROP: to initialize the //EKYMQST status file of an MQ-ASYNC Capture system.
- INIT VLF: to initialize or re-initialize the VLF class used by the MQ-ASYNC Capture system
- ESTOP DPROP: for an emergency stop of all capture activities of an MQ-ASYNC Capture system
- RESET DPROP: to reset the emergency stop status of all capture activities of an MQ-ASYNC Capture system
- DISPLAY STATUS: to show whether an MQ-ASYNC Capture system is emergency stopped

The Capture system Utility executes as a standard (non-IMS) job.

When used to create event markers, the CUT uses MQSeries services; The MQSeries Queue Manager and the MQSeries Queues identified in the //EKYTRANS file must be available for its use.

The following topics provide additional information:
- “Input and Output”
- “JCL to Execute the CUT” on page 396
- “Control statements of the CUT” on page 397
- “CUT Return Codes” on page 400

Input and Output

The input to the CUT are:
- the //CUTIN data set, which contains the utility control statements
- the //EKYIN data set
- the //EKYMQST status file
- the //EKYTRANS transmission specification file (this file is only needed when the CUT is used to create event markers)

The output from the CUT depends on which CUT function has been used:
- Changed control information in the //EKYMQST status file
- The (re-)initialized VLF class used by the DPROP Capture system
- The //CUTPRINT, //EKYPRINT, and //EKYTRACE files.
- Audit Trail records

JCL to Execute the CUT

```
//CUT      EXEC  PGM=EKYI900X
//STEPLIB  DD    DSN=DPROP.EKYRESLB,DISP=SHR
//         DD    ......
//EKYRESLB DD    DSN=DPROP.EKYRESLB,DISP=SHR
//CUTPRINT DD    SYSOUT=*
//EKYPRINT DD    SYSOUT=*
//EKYTRACE DD    SYSOUT=*
//SYSUDUMP DD    SYSOUT=*
//EKYMQST  DD    DSN=DPROP.MQST(RPE2),DISP=SHR
//EKYTRANS DD    DSN=DPROP.TRANS(RPE2),DISP=SHR
//CUTIN    DD    *
EM ID=ENDDAY , PRSTREAM=(STREAM1,STREAMB) ;
/*
```

Figure 279. JCL to Execute the Capture system Utility (CUT)

The JCL DD statements of the Capture system utility are explained below:

//STEPLIB This DD statement provides access to the Load Module Library containing the IMS DPROP Modules. When the CUT is used to create an event marker, //STEPLIB is also used to provide access to the Load Module Library containing MQSeries modules.

//EKYRESLB This DD statement is common to many IMS DPROP components and is described in detail in "//EKYRESLB DD Statement" on page 4.

//CUTPRINT Contains the print output from the CUT. The print output includes a list of all CUT control statements, confirmation that they have been executed, and a list of CUT error messages.

//EKYPRINT Contains the print output from other IMS DPROP components that are called internally by the CUT.

//EKYTRACE Contains detailed trace information that is created when IMS DPROP encounters problems. Trace information is also created if explicitly requested with TRACE Control statements of the //EKYIN data set.

//EKYMQST Provides access to the status file (typically, this is a PDS member) of the IMS DPROP Capture system.

Note: when the CUT is used to initialize the status file with an INIT DPROP=*xxxxx* control statement, the //EKYMQST DD Statement must look as follows:

```
//EKYMQST   DD    DSN=DPROP.MQST(MEMBER2),DISP=(,CATLG),
//          UNIT=SYSDA,SPACE=(TRK,(1,1,1))
```

Figure 280. Format of the //EKYMQST Statement when using an INIT DPROP control statement

//EKYTRANS Provides access to the transmission specification file of the IMS DPROP Capture system. These transmission specifications are needed by the CUT when it processes an EM (event marker) control statement. The transmission specifications identify which

MQSeries Queue Manager and which MQSeries Queues to use for the transmission of the event marker messages to the Apply programs.

By default, IMS DPROP allocates this file dynamically. If default allocation is not appropriate, the user of the EM control statements of the CUT must provide an //EKYTRANS DD Statement in the JCL of the CUT.

//CUTIN This is the file that contains the utility control statements for the CUT.

Control statements of the CUT

This topic describes the control statements of the CUT. The control statements are provided in the file described by the //CUTIN DD statement.

DISPLAY STATUS

Use the DISPLAY STATUS control statement to show whether an MQ-ASYNC Capture system is emergency stopped.

Syntax of DISPLAY STATUS control statement

```
►►──DISPLAY STATUS──;──────────────────────────────────────────────►◄
```

DISPLAY STATUS
Shows whether the IMS DPROP Capture system whose status file is defined by the //EKYMQST JCL DD statement of the CUT is emergency stopped.

EM: create an Event Marker

Use the EM control statement to create event marker MQSeries messages that will identify a Source system Event (a Source system point-in-time) to the Apply Programs.

When an Apply program reads the MQSeries message containing the event marker, it will record the event marker in the MQ-ASYNC EM table and issue a WTO. Then, depending on STOPAT control statement specifications in its //APPLYIN file, the Apply program will either stop or continue processing.

The syntax of the CUT EM control statement is shown in the following syntax diagram.

Syntax of EM control statement

```
                  ┌─,ALL──────────────────────────────┐
                  │                  ┌─;──────────────┐│
                  │                  ▼                │└│
►►──EM──ID=ID─────┴─,PRSTREAM=(──────┴─,──prstream_name──┴─)──────────►◄
```

ID= *ID*
Assigns an identification to the event. The specified *ID* must be an 8-byte alphanumeric value; the first byte must be alphabetic.

If a STOPAT= control statement of the Apply program refers to this *ID* on the Target system, the Apply program stops when reading the MQSeries containing the event marker.

ALL
The ALL keyword specifies that an MQSeries message containing the event marker be included in ALL propagation data streams defined in //EKYTRANS.

PRSTREAM=(prstream_name,.....)
Specifies that an MQSeries message containing the event marker be included in each propagation data stream identified on this keyword.

Usage Note:

1. You might want to define a STOPAT control statement specifying an agreed-to *ID* and leave it in the //APPLYIN (for example, OPERATOR). This allows the Source system operators to stop the Apply program at any Source system point-in-time, without requiring changes in the //APPLYIN control statements of the Apply program.
2. When processing an EM control statement, the CUT must have access to the //EKYTRANS transmission specification file used by the EKYMQCAP IMS Data Capture exit routine. By default, IMS DPROP allocates this file dynamically. If default allocation is not appropriate, the user of the CUT must provide an //EKYTRANS DD Statement in the JCL of the CUT.

   The CUT uses the MQSeries Queue Manager, the PRSTREAMs, and the MQSeries Output Queues defined in //EKYTRANS to understand which MQSeries Queue(s) to use to send MQSeries messages containing the event marker to the Apply program.
3. When the IMS DPROP System is emergency stopped, IMS DPROP does not attempt any MQSeries operations; therefore, it does not send any MQSeries event marker message.

ESTOP DPROP

Use the ESTOP control statement to perform an emergency stop of all capture activities of an IMS DPROP Capture system. It might take a few seconds for the Capture exit routine to be notified of the emergency stop. Also, the subsequent stopping of the capture activities is not simultaneous in the various IMS online and IMS batch regions.

Syntax of ESTOP DPROP control statement

```
►►──ESTOP DPROP──;──────────────────────────────────────────────────►◄
```

ESTOP DPROP
Performs an emergency stop of the capture activities of that IMS DPROP Capture system whose status file is defined by the //EKYMQST JCL DD statement of the CUT.

INIT DPROP=dpropname

This control statement is used to initialize the //EKYMQST status file of an IMS DPROP Capture system.

Typically, execution of the CUT with an INIT DPROP control statement is done as part of the Installation Verification Process (IVP) of IMS DPROP. It can also be used to recreate a lost or damaged status file.

If you use an INIT DPROP control statement, it must be the only CUT control statement in the //CUTIN file.

Syntax of INIT DPROP control statement

```
►►──INIT──DPROP=dpropname──┬──────────────┬──;──────────────────────────────►◄
                           └─,ESTOPPED────┘
```

INIT DPROP= *dpropname*
Initializes the status file of the IMS DPROP Capture system you identify. The specified *dpropname* is the name of the IMS DPROP Capture system that your installation has defined at DPROPGEN time.

ESTOPPED
This optional keyword requests that the IMS DPROP Capture system be initialized in stopped status.

To activate the capture activities of the IMS DPROP System, you must eventually run the CUT with a RESET DPROP control statement.

If you use an INIT DPROP control statement, the JCL of the CUT must include an //EKYMQST DD statement with the following format:

```
//EKYMQST   DD    DSN= DPROP.MQST(MEMBER2),DISP=(,CATLG),
//          UNIT=SYSDA,SPACE=(TRK,(1,1,1))
```

Figure 281. Format of the //EKYMQST Statement when using an INIT DPROP control statement

Replace *DPROP.MQST(MEMBER2)* with the name of the PDS and the name of the PDS member that has been specified during the generation of the IMS DPROP Capture system.

INIT VLF

This control statement is used to initialize or re-initialize the VLF class used by an IMS DPROP Capture system.

You can use the INIT VLF control statement after a software error created incorrect objects in VLF.

Syntax of INIT VLF control statement

```
►►──INIT VLF──;──────────────────────────────────────────────────────────────►◄
```

INIT VLF
Initializes or re-initializes the VLF class used by that IMS DPROP Capture system whose status file is defined by the //EKYMQST JCL DD statement of the CUT.

RESET DPROP

Use the RESET control statement to reset the emergency stopped status of the capture activities of an IMS DPROP Capture system. It might take a few seconds

for the Capture exit routine to be notified of the reset of the stopped status. Also, the subsequent resuming of the capture activities is not simultaneous in the various IMS online and IMS batch regions.

Syntax of RESET DPROP control statement

```
►►──RESET DPROP──;──────────────────────────────────────────────────►◄
```

RESET DPROP
Resets the emergency stopped status of that IMS DPROP Capture system whose status file is defined by the //EKYMQST JCL DD Statement of CUT.

CUT Return Codes

The CUT provides the following return codes:

0 All requested operations have been executed successfully.

4 Warning messages have been issued, for example because the IMS DPROP Capture system is in emergency stopped status.

8 Error messages have been issued when processing one or more control statements. After issuing the error messages, the CUT attempts to process the next control statement.

12 (or greater)
Error messages have been issued when processing a control statement. After issuing the error messages, the CUT terminates without attempting to process the next control statements.

Chapter 17. Status Change Utility (SCU) for MQ-ASYNC

This topic describes the use of the Status Change Utility (SCU) for an Asynchronous, MQ-based propagation (MQ-ASYNC). The SCU can also be used for Asynchronous log-based propagation (LOG-ASYNC) and for synchronous propagation; use of the SCU for LOG-ASYNC and for a synchronous propagation are described in Chapter 8, “Status Change Utility (SCU) for LOG-ASYNC ,” on page 213 and in Chapter 7, “Status Change Utility (SCU) for Synchronous Propagation ,” on page 175, respectively.

For MQ-ASYNC, use the Status Change utility (SCU) to:
- Set and reset the read-only status of full-function IMS databases
- Change the error option of propagation requests.
- Initialize the master table of the IMS DPROP directory.
- Build or initialize the VLF objects containing IMS DPROP control information used by the Apply program.
- Display the amount of virtual storage required by the VLF objects used by the Apply program

Recommendation: You are encouraged to implement DBRC share control, and to register the propagated IMS databases used for production. This will enable the SCU to support full-function IMS databases in read-only mode.

The following topics provide additional information:
- “Input and Output”
- “JCL Requirements ” on page 402
- “Use of the SCU for MQ-ASYNC ” on page 405
- “Control statements for MQ-ASYNC Propagation ” on page 406
- “Output Messages and Return Codes” on page 411

Input and Output

The input to the SCU is:
- The //SCUIN data set, which contains the control statements
- The //SCUPLAN data set
- Operator replies to WTORs

  If the SCU is in a wait state because of conflicting database authorizations, the operator can use WTOR to:
  - Enter LIST to list the conflicting authorizations.
  - Enter TERM to terminate execution of the SCU.
- The //EKYIN data set

The output from the SCU is:
- Changed control information:
  - in the DPROP directory tables
  - The virtual lookaside facility (VLF) copy of IMS DPROP information
  - The access mode of IMS databases in RECON data sets
- The //EKYTRACE, //EKYWTO, //EKYSNAP and //EKYLOG files
- Audit trail records.

JCL Requirements

This topic contains samples of the JCL used to executes the SCU.

EKYUSCUP—JCL to Execute the SCU

Figure 283 describes the sample JCL procedure EKYUSCUP used to execute the SCU.

```
//*-------------------------------------------------------------------*
//*    PROCEDURE EKYUSCUP TO EXECUTE THE SCU                          *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************

//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S NEED:          *
//*  -------------------------------------------------------          *
//*                                                                   *
//*  1111111   - DPROP RESLIB HLQ(S)                                  *
//*                                                                   *
//*-------------------------------------------------------------------*
//EKYUSCU      PROC EKYPREF=1111111
//*-------------------------------------------------------------------*
//SCU          EXEC     PGM=EKYSCU00
//*******************************************************************
//* NOTE: IN ASYNCH MODE, A STATUS FILE IS NOT REQUIRED AND THE       *
//* FOLLOWING DD STATEMENT SHOULD BE REMOVED.                         *
//*                                                                   *
//EKYSTATF  DD   DSN=&EKYPREF..EKYSTATF,DISP=SHR
//*******************************************************************
//*
//EKYRESLB     DD       DISP=SHR,
//             DSN=&EKYPREF..EKYRESLB
//SCUPRINT     DD       SYSOUT=*
//SYSPRINT     DD       DSN=&&SYSPRINT,UNIT=SYSDA,SPACE=(CYL,1)
//SYSIN        DD       DSN=&&SYSIN,UNIT=SYSDA,SPACE=(CYL,1)
//*-------------------------------------------------------------------*
//*******************************************************************
//*-----------------------------------------------------------------
//*       ADD THE RECON DD STATEMENTS HERE IF YOU ARE NOT USING
//*       DYNAMIC ALLOCATION
//*-----------------------------------------------------------------
//*       &RECON1   = --+
//*       &RECON2   =   '  -  DBRC RECON DATASETS
//*       &RECON3   = --+
//*
//*******************************************************************
//*-----------------------------------------------------------------
```

Figure 283. Sample JCL Procedure EKYUSCUP

Before you submit the JCL, replace 1111111 with IMS DPROP RESLIB high-level qualifiers.

The DD statement //EKYRESLB is common to many IMS DPROP components and is described in detail in "//EKYRESLB DD Statement" on page 4. The remaining DD statements used in the EKYUSCUP JCL are:

//EKYSTATF DD statement
When using the SCU for MQ-ASYNC, remove this DD statement from the EKYUSCUP procedure.

//SCUPRINT
Contains the print output from the SCU. The print output includes a list of all SCU control statements, confirmation that they have been executed, and a list of SCU error messages.

//SYSPRINT
Contains the print output of the IMS DBRC utility called by the SCU.

This DD statement must be allocated to a temporary data set and *not* allocated as a SYSOUT file.

//SYSIN
Contains input commands generated by the SCU to the IMS DBRC utility. The SCU calls the IMS DBRC utility.

This DD statement must be allocated to a temporary data set and *not* allocated as a SYSIN file.

For the processing of READON and READOFF control statements, the SCU uses DBRC and therefore requires access to the DBRC RECON data sets. If you use DBRC without dynamic allocation, you must add the following DD statements to the JCL.

```
//RECON1 DD DSN=recon1,DISP=SHR
//RECON2 DD DSN=recon2,DISP=SHR
//RECON3 DD DSN=recon3,DISP=SHR
```

EKYUSCUJ—JCL to Execute EKYUSCUP

Figure 284 on page 404 describes EKYUSCUJ, the JCL used to call the SCU.

```
//JJJJJJJ  JOB  *** VALID JOB CARD DETAILS ***
//*-------------------------------------------------------------------*
//*    JOB TO EXECUTE THE SCU PROCEDURE (EKYUSCUP)                    *
//*                                                                   *
//*-------------------------------------------------------------------*
//*******************************************************************
//**                                                                 *
//**        LICENSED MATERIALS - PROPERTY OF IBM                     *
//**                                                                 *
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                   *
//**                                                                 *
//**                                                                 *
//**        SEE COPYRIGHT INSTRUCTIONS                               *
//**                                                                 *
//*******************************************************************
//*-------------------------------------------------------------------*
//*                                                                   *
//*  ADAPT THE FOLLOWING VALUES TO YOUR INSTALLATION'S NEED:          *
//*  -------------------------------------------------------          *
//*                                                                   *
//*  JJJJJJJ   - JOB NAME                                             *
//*  1111111   - SCU CONTROL STATEMENT(S) AS REQUIRED                 *
//*  2222222   - DB2 SUBSYSTEM NAME                                   *
//*  3333333   - NAME OF THE SCU PLAN                                 *
//*                                                                   *
//*-------------------------------------------------------------------*
//EKYUSCU      EXEC EKYUSCUP
//SCUIN     DD   *
1111111
/*
//SCUPLAN   DD   *
SYSTEM=2222222,PLAN=3333333;
//*-------------------------------------------------------------------*
```

Figure 284. EKYUSCUJ: Sample JCL to Run the SCU

//SCUPLAN DD statement

Identifies the file that contains a control statement that identifies the name of the DB2 plan that executes the SCU. The //SCUPLAN file is input to the SCU.

The SCU uses the CALL ATTACH FACILITY (CAF) to connect to DB2. It can be executed as a standard MVS/ESA batch job step or under TSO. When the SCU runs under TSO, it uses CAF, not the TSO attach facility.

In a multisite environment (when the MQ-ASYNC Capture component and MQ-ASYNC Apply program run on different sites), when the SCU runs only on the IMS side, you do not need to specify the name of the DB2 plan because the SCU will not need to access DB2.

Figure 285 shows the syntax of the //SCUPLAN file.

```
►►──PLAN=plan_name──┬──────────────────────┬──;──────────────────────────►◄
                    └──,SYSTEM=system_name─┘
```

Figure 285. The //SCUPLAN file

PLAN= *plan_name*

Names the DB2 plan that executes the SCU.

SYSTEM= *system_name*
Identifies the DB2 subsystem. If you do not use this keyword, the SCU uses the default DB2 subsystem identifier in module DSNHDECP.

Figure 286 shows two examples of the //SCUPLAN control statement. The first example shows that DB2 plan TEST01 executes the SCU on the subsystem identified in DSNHDECP. The second example shows that DB2 plan TEST01 executes the SCU on system TEST02.

```
PLAN=TEST01
PLAN=TEST02,SYSTEM=TEST02;
```

Figure 286. //SCUPLAN control statement Examples

//SCUIN
Contains the stream of SCU control statements.

Use of the SCU for MQ-ASYNC

With MQ-ASYNC, the SCU is used to:
- Set and reset the read-only status of full-function IMS databases
- Change the error option of propagation requests.
- Initialize the master table of the IMS DPROP directory.
- Build or initialize the VLF objects containing IMS DPROP control information used by the Apply program.
- Display the amount of virtual storage required by the VLF objects used by the Apply program

Setting and Resetting the Read-Only Status of IMS Databases

IMS DPROP and the SCU enable IMS databases to be set in read-only mode and to wait until all IMS subsystems release their database update authorization.

This allows the IMS databases to quiesce, when you want to reflect a clean Source system point-in-time in the target DB2 tables . This can be used to prevent IMS database updates during an IMS database extract.

This SCU support is provided only for registered, full-function databases if share control is in effect. If share control is in effect, you can use the SCU to set one of the following two database states:
- Read-only (READON)
- Read-only-off (READOFF)

IMS records these states in the DBRC RECON data sets.

When you call the SCU to set read-only, it calls the IMS DBRC utility with the CHANGE.DB READON command. The SCU returns with a zero return code when all IMS subsystems release their database authorization. You can then assume that no IMS subsystems are updating the database and it is safe to extract the database.

To set read-only-off for a database, the SCU calls the IMS DBRC utility with the CHANGE.DB READOFF command. When the database state is returned to read-only-off, the IMS subsystems can again update the databases.

Control statements for MQ-ASYNC Propagation

Multi-site Environments

In a multi-site environment, some control statements are specific to the Source system or Target system.

Source Site control statements

Source site control statements have no dependency on DB2. The following Source site statements control whether the IMS subsystems have update access to the databases:

- READON
- READOFF

Target Site control statements

The IMS DPROP directory tables exist on the Target site. The following control statements require access to the IMS DPROP directory tables and require that the SCU run on the Target site are:

- DISPLAY
- ERROPT
- INIT

Syntax of the SCU control statement

Figure 287 describes the syntax of the PR Identification block used in many SCU control statements. For a complete description of the propagation request identification block, see “The Propagation Request Identification Block ” on page 225. The syntax is provided here for reference.

control statements

```
►►──Statement name──────────────────────────────────────────────────────►◄

                  ┌──,──┐
►►──┬─PR=(──────┬─pr_id─┬──)──────────────────────────────────────┬──;──►◄
    │           └───────┘                                         │
    │                  ┌──,──┐                                    │
    ├─PRSET=(─────┬─prset_id─┬──)──────────────────────────────────┤
    │             └──────────┘                                    │
    │                  ┌──,──┐                                    │
    ├─DBD=(───────┬─dbd_name─┬──)──────────────────────────────────┤
    │             └──────────┘                                    │
    │                                  ┌──,──┐        ┌──,──┐     │
    ├─DBD=(─dbd_name─),SEG=(──────┬─seg_name─┬──),PRSET=(──────┬─prset_id─┬──)─┤
    │                             └──────────┘        └─────────┘           │
    └─ALL──────────────────────────────────────────────────────────────────┘
```

Figure 287. Syntax of the PR Identification Block used in SCU

DISPLAY

Use the DISPLAY control statement to show the total amount of virtual storage required by the virtual lookaside facility (VLF) objects used by the Apply Program of this IMS DPROP system.

You can use the information on virtual storage to adjust the MAXVIRT value for the VLF class in the COFVLFxx member. For more information on the MAXVIRT value, see *IMS DPROP Installation Guide*.

Syntax of the DISPLAY control statement

Figure 288 shows the syntax of the DISPLAY control statement.

```
►►──DISPLAY──STATUS──;──────────────────────────────────────────────►◄
```

Figure 288. The DISPLAY control statement

DISPLAY
Displays the operating characteristics of the IMS DPROP system.

STATUS
Checks the status of the IMS DPROP system.

Example of the DISPLAY control statement

Figure 289 shows an example of the DISPLAY control statement.

```
DISPLAY STATUS
```

Figure 289. DISPLAY control statement Example

Example of a DISPLAY control statement Response

Figure 290 shows the response to a DISPLAY control statement.

DISPLAY (REQUIRED)
Displays the operating characteristics of the IMS DPROP system.

STATUS (REQUIRED)
Checks the status of the IMS DPROP system.

```
EKYS467I DPROP SYSTEM STATUS: PROPAGATION IS ACTIVE   DPROP SYSTEM NAME: T003815
         DPROP SYSTEM LEVEL: 110

EKYS468I VLF OBJECT SIZE: 44928 BYTES
         VLF CLASS NAME: PM3
```

Figure 290. DISPLAY Response Example

ERROPT

Use the ERROPT statement to change the error option of a propagation request. The options are IGNORE and BACKOUT.

Syntax of the ERROPT control statement

Figure 291 on page 408 shows the syntax of the ERROPT statement. For a description of the propagation request identification block, see “The Propagation Request Identification Block ” on page 225.

```
►►─ERROPT──┬─IGNORE──┬──────────────────────────────────────────────►
           └─BACKOUT─┘

►──┬─,PR=(──┬─pr_id──┬─)──────────────────────────────────────────;──►◄
   │        ┌──,──┐
   ├─,PRSET=(──┬─prset_id──┬─)──────────────────────────────────────┤
   │        ┌──,──┐
   ├─,DBD=(──┬─dbd_name──┬─)──┬──────────────────────────────────┤
   │        ┌──,──┐          │ ┌──,──┐                            │
   │        │     │          └─,PRSET=(──┬─prset_id──┬─)──┘        │
   │        ┌──,──┐
   ├─,DBD=dbd_name,SEG=(──┬─seg_name──┬─)──┬──────────────────┤
   │                      ┌──,──┐          │ ┌──,──┐            │
   │                      │     │          └─,PRSET=(──┬─prset_id──┬─)──┘
   └─,ALL──────────────────────────────────────────────────────┘
```

Figure 291. Syntax of the ERROPT control statement

IGNORE | BACKOUT

Use IGNORE to disregard errors issued when propagation cannot be successfully completed.

Use BACKOUT to back out the changes made since the last commit point when errors occur.

Examples of the ERROPT Statement

Figure 292 shows an example of ERROPT with the IGNORE option, and with the BACKOUT option.

```
ERROPT IGNORE,DBD=SKILLS
ERROPT BACKOUT,DBD=SKILLS,SEG=ROOT;
```

Figure 292. ERROPT Examples

INIT

Use the INIT control statement to initialize IMS DPROP control information.

Syntax of the INIT control statement

Figure 293 on page 409 shows the syntax of the INIT control statement.

```
►►──INIT──────────────────────────────────────────────────────────────►◄

►►──┬─DPROP=dpropname─┬──;──────────────────────────────────────────────►◄
    └─VLF─────────────┘
```

Figure 293. The INIT control statement

INIT
Initializes the IMS DPROP directory.

DPROP= *dpropname*
Specifies the name of the IMS DPROP directory for the IMS DPROP system you want to initialize. *dpropname* is defined at installation time.

When you issue the INIT DPROP statement, the SCU verifies that the master table in the IMS DPROP directory is empty. If it is not empty, the SCU writes an error message and stops further processing.

If the directory is empty, the SCU inserts the unique row of the master table of the IMS DPROP directory. This row contains the IMS DPROP system name provided during IMS DPROP installation, and a unique token generated by the SCU.

For more information about the master table, see “The Master Table (DPRMASTER) ” on page 143.

VLF
Builds the VLF objects, which contain IMS DPROP control information used by the Apply program.

Use this keyword to populate VLF after an IPL or after a sequence of VLF deactivation and reactivation. All objects of the VLF class belonging to the IMS DPROP system are deleted before being recreated. Use an INIT VLF statement to populate VLF rather than let the Apply program populate VLF on a demand basis. Doing so avoids ENQueue conflicts on small IMS DPROP directory tables.

You can also use an INIT VLF after a software error created incorrect objects in VLF, or after recovering the IMS DPROP directory tables.

Examples of the INIT control statement

The example in Figure 294 initializes IMS DPROP directory TEST01.

```
INIT DPROP=TEST01
```

Figure 294. INIT Example

READOFF

Use the READOFF control statement to turn off DBRC read-only mode for an IMS full-function database. The READOFF control statement supports full-function IMS databases only if DBRC share control is active and the database is registered in DBRC.

The READOFF control statement resets an IMS database after it was set to read-only to allow database quiesce event markers to be created.

READOFF control statement

Figure 295 shows the syntax of the READOFF control statement.

```
                    ┌─,──────────┐
                    ▼            │
►►──READOFF──DBD=(──dbd_name──────)──;──────────────────────────────────►◄
```

Figure 295. The Syntax of the READOFF control statement

READOFF
Resets the read-only mode of IMS databases.

DBD=(*database1,database2,...* **)**
Names the databases for which read-only mode is reset. Specify the names of a physical IMS databases.

The SCU terminates processing if an error occurs while processing a control statement. To prevent your IMS databases from being left in a read-only state, provide a separate JCL step to set READOFF mode if an SCU control statement fails. See Figure 65 on page 179 for an example of the JCL.

Examples of the READOFF control statement

The example in Figure 296 sets database SKILLDB to READOFF mode.

```
READOFF DBD=SKILLDB
```

Figure 296. READOFF Example

READON

Use the READON control statement to set an IMS full-function database to DBRC read-only mode. The READON control statement supports full-function databases only if DBRC share control is active and if the database is registered in DBRC.

The READON control statement allows database quiesce event markers to be created.

When processing a READON control statement, the SCU waits until the updating IMS subsystems have released their database update authority. This is in contrast to the DBRC CHANGE.DB READON commands: DBRC does not wait until IMS systems release their update authority before returning. For more information on DBRC share control, refer to *IMS/ESA Operations Guide*.

Syntax of the READON control statement

Figure 297 on page 411 shows the syntax of the READON control statement.

```
                  ┌─,──────────┐
                  ▼            │
►►──READON──DBD=(──dbd_name──┘──)──;──────────────────────────────────►◄
```

Figure 297. Syntax of the READON control statement

READON
Sets the read-only mode of a physical IMS databases.

DBD=(*database1,database2,...* **)**
Specifies the IMS database or list of databases to set to read-only mode.

Example of the READON control statement

The example in Figure 298 sets the databases names SKILLDB and INVENTRY to READON mode.

```
READON DBD=(SKILLDB,INVENTRY)
```

Figure 298. READON Example

Output Messages and Return Codes

Error messages for the SCU begin with EKYS. For more information on SCU messages and codes, refer to*IMS DPROP Messages and Codes*.

The Status Change utility provides the following return codes:

Code	Meaning
0	All requested changes have executed successfully.
4	Warning messages were issued for one or multiple of the following reasons: • The SCU processes READON of READOFF control statements, and an IMS database could not be set to the requested state because: – DBRC has not been defined for share control. – The IMS database is not registered in RECON. – The IMS database is a DEDB.
8	Some or all of the requested changes were not made. This return code occurs when: • There is no valid control statement in //SCUIN data set. • The operator replies TERM to an outstanding WTOR console message.
16	A severe error was detected. This return code occurs when no valid control statement is found in //SCUPLAN data set (when //SCUPLAN is required), or when there is a termination error message.

Part 7. Consistency Check Utility (CCU)

Chapter 18. Consistency Check Utility (CCU)

The Consistency Check utility (CCU) checks IMS and DB2 data to ensure that there is no redundant data, and that the data is consistent between IMS and DB2 databases.

The CCU reports and describes data inconsistencies, and creates repair statements to repair the IMS database and DB2 tables if mismatches occurred.

You can perform consistency checks at regular intervals, or as needed. Times when you might use the CCU are:

- After defining a new propagation request, or changing an existing propagation request
- After a database reload on the IMS or DB2 side, or both
- After a data propagation failure
- After receiving a suspended or deactivated propagation status
- After database repair actions with PROP OFF
- After application or operator handling errors

You should run the CCU after the IMS updates are propagated to the DB2 tables, so both data copies have the same data content.

In order to run the CCU you must:

- Have DB2 up and running.
- Have the SELECT privilege for all DB2 tables that the CCU checks.
- Have the required authorization to submit the CCU. For more information on authorization, see the appropriate Administrators Guide for your propagation mode.

The CCU runs in an IMS BMP region. The CCU supports checking of up to 256 PRIDs per job submission.

The CCU does not support user-mapping PRTYPEs (PRTYPE=U).

There are two ways you can run the CCU:

- With JCL and CCU control statements, which are described in this topic
- With the IMS DPROP-provided ISPF panel application, which is described in Chapter 20, “CCU Front End ,” on page 523

Chapter 19, “Interpreting CCU Reports,” on page 467 gives examples of CCU reports that the CCU generates, and information on how to interpret the reports.

The following topics provide additional information:

- “The Read and Compare Phase ” on page 450
- “The Error Location Phase ” on page 457
- “Repair Data Set Generation ” on page 459
- “The Check control statement ” on page 460

Phases of the CCU

Dividing the CCU into separate phases allows you to run some job steps in parallel and some independently. The phases of the CCU are:

Initialization

Description of the initialization phase begins on page 447.

The initialization phase is the first step in a CCU job stream. In the initialization phase, you define what data to check, what technique to use to check the data, and what processing options to use for the remaining CCU phases. The CCU initialization phase creates a PSB load module to retrieve the IMS data and a control data set (//CCUCDS) containing more information required for subsequent CCU phases (such as the SQL SELECT statements for the DB2 table retrieval process).

Read and compare

Description of the read and compare phases begins on page 450.

During the read and compare phase the CCU retrieves segments from IMS and rows from DB2 tables. If you use the hashing technique, the CCU creates files that represent hash sums of the data. Then it compares the hash sum files to find whether they are equal. If you use the direct technique, the CCU reads and compares the IMS segments and the DB2 rows in one step.

Error location

Description of the error location phase begins on page 457.

During the error location phase the CCU verifies data mismatches between IMS and DB2, generates error messages, and generates repair statements depending on what keywords you specified on control statements.

Space Requirements for CCU Data Sets

The CCU data sets require processing space in your MVS system. Table 37 provides guidelines on how to calculate the space requirements for the data sets created by CCU programs.

Table 37. Space Requirements in BLOCKS for CCU Data Sets

File	BLKSIZE	Rule
//CCUDBOUT	8196	((((386 + f) d) / 8196) + 1) per PRID
//CCUHSUM1 //CCUHSUM2	32760	(2a (b + 8) / 32760) + 1
//CCUHASH1 //CCUHASH2	7472	(16c / 7472) + 1
//CCUKEY1 //CCUKEY2	32768	((d (e + 6) / 32768) + 1) per PRID
//CCUSORTS	800	(80a / 800) + 1
//CCUMSMTC		no rule

Table 37. Space Requirements in BLOCKS for CCU Data Sets (continued)

File	BLKSIZE	Rule
//CCUDERR		no rule
//CCUIERR		no rule

Where:
a = number of propagation request IDs to be checked
b = length of longest DB2 row, or mapped IMS segment type, to be checked
c = number of DB2 rows (total over all DB2 tables) to be checked
d = number of DB2 rows per propagation request ID to be checked
e = length of DB2 primary key
f = length of DB2 row, or mapped IMS segment type, to be checked

When the CCU uses HD unload copies of the IMS database as input, the CCU allocates sufficient storage to build a copy of the segments for error location. See Table 37 on page 418. If the amount of storage required for CCUDBOUT is unacceptable, the following procedure can reduce the amount of space required:

1. Run the CCU up to the compare phase, not including the error location phase, specifying CCUDBOUT DD DUMMY in the compare phase.
2. Examine any mismatches that arise in the compare phase and make a note of the DBDs and propagation requests involved.
3. Rerun the CCU with error location and with a correctly specified CCUDBOUT DD statement for only the DBDs and propagation requests causing mismatches.

The CCU allocates storage based on the size of the largest segment to propagate. Therefore, this procedure reduces the size of the CCUDBOUT data set only if the largest segment propagated is not involved in any of the propagation requests causing mismatches.

Using the CCU in an Asynchronous IMS DPROP System

In an asynchronous system you should submit the CCU after all relevant IMS database updates have been propagated to the DB2 tables. Otherwise, the CCU creates a large data inconsistency report.

Asynchronous propagation does not support the direct technique, you must use the hashing method for the read and compare phase.

You can use either an IMS HD unload file, a DEDB unload file or the database as IMS input.

You can run the CCU whether IMS and DB2 reside on the same or different MVS systems.

If IMS and DB2 Data Reside on the Same System

If both the IMS and DB2 data reside on the same MVS system, you can run the CCU using either the IMS database or the HD unload file as input. Table 38 on page 420 describes some advantages and disadvantages.

Table 38. Input to CCU: Advantages and Disadvantages

Input from	Advantages	Disadvantages
IMS Database	You can submit the CCU without any job modifications if you are using a synchronous system.	The IMS database should remain in read-only status until the CCU has completed its job. If the IMS database is not in read-only status, the CCU might use IMS data that has not yet been properly processed by the Receiver or Apply.
HD unload file	You can start updates to the IMS database after the HD unload file is created.	• You must unload the IMS database to create an HD unload file. • The CCU cannot: – Check whether or not the HD unload file corresponds to the current physical DBD definition. – Compare the DBD version ID with the version ID stored in the IMS DPROP directory because the HD unload file does not contain a version ID.
DEDB unload file	Same as an HD unload file	Same as an HD unload file

If IMS and DB2 Data Reside on Different Systems

If the IMS data is located on a different MVS system than the DB2 data, you must use one of the following utilities to submit the IMS read phase:

- The HD unload file created by the HD Reorganization Unload utility, DFSURGU0.
- The HSSR Database Unload utility, or equivalent
- The DEDB unload file created by the IMS DPROP DEDB unload program, EKYU000X. See *IMS/ESA Utilities Reference: System* and *IMS/ESA Utilities Reference: Database Manager* for complete information on DFSURGU0.

How to Use the HD Unload File

In an asynchronous system, the CCU uses an unload file that was created by the IMS HD unload utility, DFSURGU0, to compare IMS and DB2 data. To create an unload file follow these steps:

1. Submit an HD unload job for your IMS database using DFSURGU0, or an equivalent program.

   However, if your IMS database format is either SHISAM or DEDB, you cannot use DFSURGU0 to unload the file. For SHISAM you must submit your own application program that creates an unload file in a format equivalent to the format created by DFSURGU0. For DEDB, you can run EKYU000X to create an unload file in the required format.
2. After you have created the unload file, and if IMS and DB2 are on remote systems, you must transmit the following files to the DB2 system:
   - The generated unload file
   - A copy of the IMS DBD load module, with its logical related IMS DBDs, if applicable
3. After you transmit the files to the DB2 system, and the Receiver or Apply Component has completed its job, you can submit the CCU.

4. Add the following statements to the JCL for the HD unload file:

   For the read phase
   Replace the IMS database DD statements with both the //CCUDBIN DD statement and the name of the HD unload file.

   For the compare phase
   Add the //CCUKEY1 and //CCUKEY2 files.

   For the error location phase
   Replace the IMS database DD statements with the //CCUDBOUT statement.

5. Do not start a new Apply or Receiver cycle until the CCU fully completes its job. If you do, the CCU can generate unpredictable DB2 repair statements. For more information on the Receiver or Apply, see the appropriate Administrators Guide for your propagation mode.

When you use an HD unload file as input, you do not need a PSB to execute the CCU. Omit the PSBNAME= keyword of the CCU CHECK statement to prevent the CCU from creating a PSB. Figure 305 on page 439 shows the sample JCL for EKYUCCHD to execute the CCU using an HD unload file as input to the read and compare phase.

The IMS read and the error location phases run under TSO/TMP.

JCL for the DB2 read and hash sum compare phases is the same as in the synchronous IMS DPROP system.

Figure 299 on page 422 is a timeline that illustrates the use of the HD unload job.

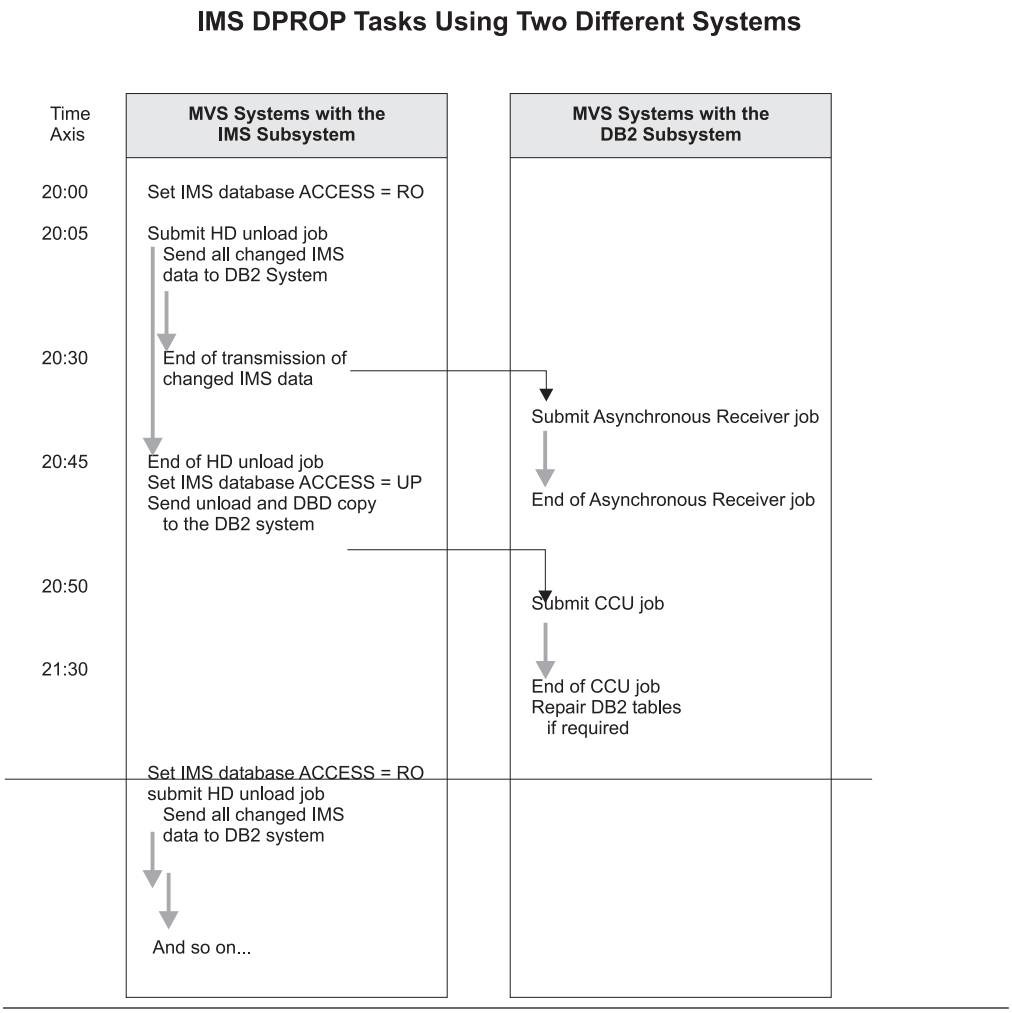

Figure 299. IMS DPROP Tasks Using 2 Different Systems

Hashing and Direct Techniques

The CCU supports two different techniques during the read and compare phase:

The hashing technique

Performs the read, hash sum compare, and compare steps separately. You must use the hashing technique in an asynchronous system. However, the hashing technique is always valid to use for either an asynchronous or synchronous system. If the retrieval sequences of the IMS data and DB2 primary key are different, use the hashing technique.

The direct technique

Performs the IMS and DB2 read and compare phases in one step. You can use the direct technique with a synchronous system only. The direct technique assumes that the IMS retrieval sequence and the DB2 primary key index sequence are the same. Different sequences lead to synchronization problems in the direct technique and cause false mismatches to be written to the mismatch indication file, CUMSMTC. If few inconsistencies exist, the direct technique requires less elapsed time than the hashing technique.

The main difference between the two techniques is that the direct technique allows the comparison to be made *as the two data copies to be compared are read*. With the hashing technique, the CCU requires use of output files and subsequent sort and compare phases.

You *must* use the hashing technique if:

- You are using a IMS DPROP asynchronous system or user asynchronous system.
- The IMS retrieval sequence does not match the DB2 primary key index sequence. For example, the two sequences seldom match when:
  - Any segment type within the generated PSB has a nonunique key field definition in the DBD (FIELD NAME=(name,SEQ,*M*)).
  - Any segment type within the generated PSB has no key field definition, but has multiple occurrences under its parent.
  - The IMS database is either HDAM or DEDB, and neither uses a sequential randomizer.
  - The IMS key sequence is different from the DB2 primary key index sequence. This can happen because IMS uses a byte-by-byte binary ascending sequence; the DB2 sequence depends on the data type and can be either ascending or descending. For example, the two sequences will not match when an IMS packed key field that can have negative values is propagated.
  - A field exit routine converts key fields so that the sequences of the original and converted fields are different.

Table 39 summarizes the use of the hashing and direct techniques.

Table 39. Summary of Hashing and Direct Techniques

Condition	Database	Hashing	Direct
IMS DPROP system is synchronous. All entity segments to be checked, using their physical path, can be retrieved in the same key sequence as the DB2 rows from the propagated DB2 tables using their DB2 primary key index.	HIDAM HISAM SHISAM HDAM DEDB	X X X X X	X X X X1 X2
At least one entity segment type to be checked, using its physical path, cannot be retrieved in the same key sequence as the DB2 rows from the propagated DB2 tables using the DB2 primary key index.	HIDAM HISAM SHISAM HDAM DEDB	X X X X X	
IMS DPROP system is LOG-ASYNC or user asynchronous	any of the above databases	X	

Notes:

1. 1 When using HDAM sequential randomizing
2. 2 When using DEDB sequential randomizing

The differences in the phases of the hashing and direct techniques are illustrated in Figure 300 on page 424.

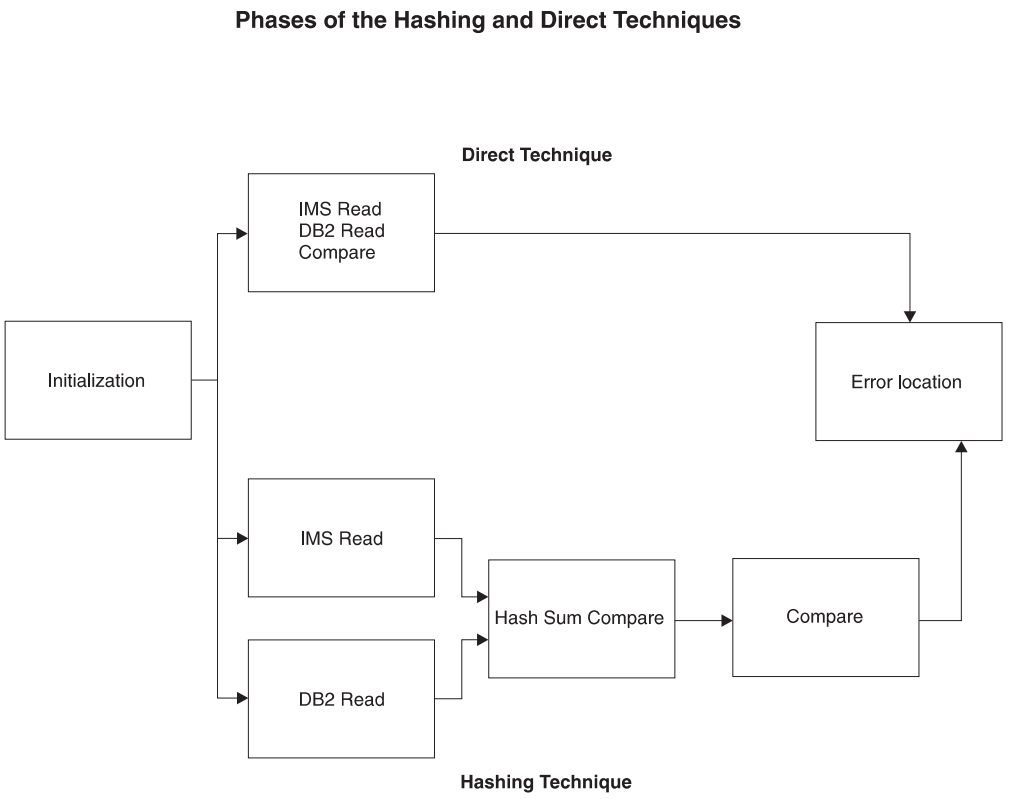

Figure 300. Phases of the Hashing and Direct Techniques

JCL for the CCU

The CCU runs as a relational application in a DL/I batch region or in an IMS BMP region.

This topic contains samples of JCL to execute the CCU using the hashing and direct techniques. The JCL samples show all three phases of CCU execution. However, only the read and compare phase let you choose between the hashing or direct technique. The JCL samples in this topic are:

EKYUCCUD Executes the CCU direct technique. A sample is in Figure 302 on page 426.

Figure 303 on page 430 is an example of the JCL for the direct technique when you used the READONLY keyword on the CHECK control statement during the initialization phase.

EKYU000X Creates a DEDB unload file. A sample is in Figure 301 on page 425.

EKYUCCUH Executes the CCU hashing technique. A sample is in Figure 304 on page 432.

EKYUCCUHD Executes the CCU hashing technique when the IMS data is on an HD unload file. A sample is in Figure 305 on page 439.

See “DD Statements ” on page 444 for descriptions of the DD statements for all the CCU job steps.

EKYU000X—JCL to Create a DEDB Unload file

Figure 301 shows the sample JCL for running EKYU000X.

```
//JJJJJJJJ JOB                                                          00000100
//********************************************************************** 00000210
//**                                                                  * 00000220
//**        LICENSED MATERIALS - PROPERTY OF IBM                      * 00000230
//**                                                                  * 00000240
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                    * 00000250
//**                                                                  * 00000260
//**                                                                  * 00000270
//**        SEE COPYRIGHT INSTRUCTIONS                                * 00000280
//**                                                                  * 00000290
//********************************************************************** 00000291
//********************************************************************** 00000292
//*                                                                   * 00000293
//*       SAMPLE JCL TO RUN THE DEDB UNLOAD UTILITY                   * 00000294
//*                                                                   * 00000295
//********************************************************************** 00000296
//S01     EXEC IMSBATCH,IMSID=IMSA,                                     00000300
//             MBR=EKYU000X,PSB=BMP001,NBA=10,OBA=5                     00000400
//EKYPRINT DD  SYSOUT=*                                                 00000500
//DBD      DD  DSN=IMSESA.DBDLIB,DISP=SHR                               00000600
//HDOUT    DD  DSN=EKY.HDOUT.FILE,UNIT=SYSDA,SPACE=(CYL,(1,1)),         00000700
//             DCB=(RECFM=VB,BLKSIZE=27998,LRECL=27994),                00000800
//             DISP=(,CATLG)                                            00000900
//SYSIN    DD  *                                                       00001000
* UNLOAD DATABASE 'TOOLS'                                              00001100
DBD=TOOLS                                                              00001200
```

Figure 301. Sample JCL for EKYU000X to Create DEDB Unload file

//EKYPRINT DD Statement
Identifies a SYSOUT data set where messages are displayed.

//DBD DD Statement
Identifies the library that contains the DBD for the DEDB to be unloaded.

//HDOUT DD Statement
Defines the output DEDB unload file. The record length must be sufficient to contain the longest record written. This will be the larger of the two following lengths:
- The longest segment length plus 41
- The number of segments multiplied by 40 plus 8

//SYSIN DD Statement
Used to input the control card which identifies the DEDB to be unloaded. The control card must be in the format DBD=*dddddddd* where *dddddddd* is the DEDB name. The control card must start in column one. A card with an asterisk in column one is treated as a comment card. If there is more than one non-comment control card, only the first one is processed.

EKYUCCUD–Sample JCL for the Direct Technique

Figure 302 on page 426 shows the sample JCL for EKYUCCUD, which is used to execute the CCU read and compare phase using the direct technique.

```
//JJJJJJJ JOB  ***  VALID JOB CARD DETAILS  ***                        00001000
//******************************************************************** 00002000
//**                                                                */ 00003000
//**        LICENSED MATERIALS - PROPERTY OF IBM                    */ 00004000
//**                                                                */ 00005000
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                  */ 00006000
//**                                                                */ 00007000
//**                                                                */ 00008000
//**        SEE COPYRIGHT INSTRUCTIONS                              */ 00009000
//**                                                                */ 00010000
//******************************************************************** 00011000
//******************************************************************** 00013000
//*                                                                 */ 00014000
//*                SAMPLE JOB STREAM FOR THE                        */ 00015000
//*             CONSISTENCY CHECK UTILITY (CCU)                     */ 00016000
//*                 - DIRECT TECHNIQUE                              */ 00017000
//*                                                                 */ 00018000
//*                                                                 */ 00019000
//*        THIS JOB CAN BE USED AS A SAMPLE JOB CONTROL STREAM      */ 00020000
//*        IF YOU HAVE NOT PLANNED TO USE THE DPROP CCU ISPF        */ 00021000
//*        SAMPLE APPLICATION.                                      */ 00022000
//*                                                                 */ 00023000
//*        FOLLOWING VARIABLES SHOULD BE ADAPTED IF YOU'RE NOT      */ 00024000
//*        USING THE DPROP INSTALLATION PROCEDURES:                 */ 00025000
//*                                                                 */ 00026000
//*         1111111    = DPROP HLQ(S)                               */ 00027000
//*         2222222    = DB2 SUBSYSTEM ID                           */ 00028000
//*         3333333    = DB2 APPLICATION PLAN FOR CCU               */ 00029000
//*                                                                 */ 00030000
//*        FOLLOWING VALUES MUST BE CHANGED FOR YOUR ENVIRONMENT    */ 00031000
//*                                                                 */ 00032000
//*         4444444   = IMS MACRO LIBRARY                           */ 00033000
//*         5555555   = IMS DBD LIBRARY                             */ 00034000
//*         6666666   = IMS DBD NAME                                */ 00035000
//*         7777777   = IMS PSB LIBRARY                             */ 00036000
//*         8888888   = IMS PSB NAME                                */ 00037000
//*                                                                 */ 00038000
//*         //DATABASE = NAME OF THE IMS DATABASE TO BE CHECKED     */ 00039000
//*         //INDEX    = NAME OF THE IMS DATABASE PRIMARY INDEX     */ 00040000
```

Figure 302. Sample JCL for EKYUCCUD for the Direct Technique (Part 1 of 4)

```
//*                                                                  */ 00041000
//*******************************************************************/ 00042000
//*                                                                  */ 00043000
//*            DATA CONSISTENCY CHECK UTILITY                        */ 00044000
//*                                                                  */ 00045000
//*            DIRECT TECHNIQUE IS BEING USED                        */ 00046000
//*                                                                  */ 00047000
//*------------------------------------------------------------------*/ 00048000
//*                                                                  */ 00049000
//*            I N I T I A L I Z A T I O N   P H A S E               */ 00050000
//*                                                                  */ 00051000
//*------------------------------------------------------------------*/ 00052000
//DELETE       EXEC     PGM=IDCAMS                                     00053000
//SYSPRINT     DD       SYSOUT=*                                       00054000
//SYSIN        DD       *                                              00055000
 DELETE '1111111.CCUCDS'                                               00056000
//*--------------------*                                               00057000
//INIT         EXEC     PGM=IKJEFT01,DYNAMNBR=30                       00058000
//EKYRESLB     DD       DISP=SHR,                                      00059000
//             DSN=1111111.EKYRESLB                                    00060000
//SYSTSPRT     DD       SYSOUT=*                                       00061000
//SYSTSIN      DD       *                                              00062000
 DSN SYSTEM(2222222)                                                   00063000
 RUN PROGRAM(EKYC000X) -                                               00064000
     PLAN(3333333)                                                     00065000
//SYSPRINT     DD       SYSOUT=*                                       00066000
//SYSIN        DD       DUMMY                                          00067000
//ASYSIN       DD       DISP=(,PASS),DSN=&&ASYSIN,UNIT=SYSDA,          00068000
//             DCB=(BLKSIZE=720,RECFM=FB),SPACE=(TRK,1)                00069000
//ASYSPRT      DD       SYSOUT=*                                       00070000
//ASYSLIB      DD       DISP=SHR,DSN=4444444                           00071000
//             DD       DISP=SHR,DSN=SYS1.MACLIB                       00072000
//ASYSLIN      DD       DISP=(,PASS),DSN=&&LKEDIN,UNIT=SYSDA,          00073000
//             DCB=(BLKSIZE=720,RECFM=FBS),SPACE=(CYL,1)               00074000
//LSYSPRT      DD       SYSOUT=*                                       00075000
//SYSLMOD      DD       DISP=SHR,DSN=7777777                           00076000
//SYSUT1       DD       UNIT=SYSDA,SPACE=(CYL,3)                       00077000
//SYSUT2       DD       UNIT=SYSDA,SPACE=(CYL,3)                       00078000
//SYSTERM      DD       SYSOUT=*                                       00079000
//CCUPRINT     DD       SYSOUT=*                                       00080000
//CCUCDS       DD       DISP=(,CATLG,DELETE),                          00081000
//             DSN=1111111.CCUCDS,                                     00082000
//             UNIT=SYSDA,SPACE=(TRK,(1,1),RLSE)                       00083000
//CCUDBD       DD       DISP=SHR,DSN=5555555                           00084000
//CCUIN        DD       *                                              00085000
CHECK DBD=6666666                                                     00086000
    , DIRECT                                                          00087000
    , PSBNAME=8888888                                                 00088000
     ;                                                                00089000
```

Figure 302. Sample JCL for EKYUCCUD for the Direct Technique (Part 2 of 4)

```
//*-----------------------------------------------------------------*/  00090000
//*                                                                 */  00091000
//*            IMS / DB2 DIRECT TECHNIQUE READ PHASE                */  00092000
//*                                                                 */  00093000
//*-----------------------------------------------------------------*/  00094000
//DELETE       EXEC     PGM=IDCAMS,COND=(8,LE,INIT)                     00095000
//SYSPRINT     DD       SYSOUT=*                                        00096000
//SYSIN        DD       *                                               00097000
 DELETE '1111111.CCUMSMTC'                                              00098000
//*--------------------*                                               00099000
//READ         EXEC     DLIBATCH,                                       00100000
//             COND=(8,LE,INIT),                                       00101000
//             PSB=8888888,                                            00102000
//             MBR=DSNMTV01                                            00103000
//G.IMS        DD       DISP=SHR,                                       00104000
//             DSN=7777777                                             00105000
//             DD       DISP=SHR,                                       00106000
//             DSN=5555555                                             00107000
//IEFRDER      DD       UNIT=SYSDA,SPACE=(CYL,(20,10))                  00108000
//DFSVSAMP     DD       *                                               00109000
1024,6            /*   Change     */                                   00110000
2048,6            /* these values */                                   00111000
4096,6            /*     as       */                                   00112000
IOBF=(4096,6,N,N) /*   required   */                                   00113000
OPTIONS,DL/I=ON,LOCK=ON,DISP=ON,SCHD=ON                                00114000
//DDOTV02      DD       DISP=(,PASS),DSN=&&DDOTV02,                     00115000
//             DCB=(LRECL=4092,BLKSIZE=4096,RECFM=VB),                 00116000
//             SPACE=(TRK,1),UNIT=SYSDA                                00117000
//DDITV02      DD       *                                               00118000
 2222222,,DSNMIN10,,R,,8888888,3333333,EKYC300X                         00119000
/*                                                                     00120000
//EKYRESLB     DD       DISP=SHR,                                       00121000
//             DSN=1111111.EKYRESLB                                    00122000
//CCUPRINT     DD       SYSOUT=*                                        00123000
//CCUCDS       DD       DISP=SHR,                                       00124000
//             DSN=1111111.CCUCDS                                      00125000
//CCUMSMTC     DD       DISP=(,CATLG),                                  00126000
//             DSN=1111111.CCUMSMTC,                                   00127000
//             UNIT=SYSDA,SPACE=(CYL,(1,10),RLSE)                      00128000
//DATABASE     DD        DISP=SHR,DSN=???????                          00129000
//INDEX        DD        DISP=SHR,DSN=???????                          00130000
//*-----------------------------------------------------------------*/  00131000
//*                                                                 */  00132000
//*            E R R O R   L O C A T I O N   P H A S E              */  00133000
//*                                                                 */  00134000
//*-----------------------------------------------------------------*/  00135000
//DELETE       EXEC     PGM=IDCAMS,COND=(8,LE,INIT)                     00136000
//SYSPRINT     DD       SYSOUT=*                                        00137000
//SYSIN        DD       *                                               00138000
 DELETE '1111111.CCUDERR' PURGE SCR                                     00139000
 DELETE '1111111.CCUIERR' PURGE SCR                                     00140000
//*--------------------*                                               00141000
//ERRORLOC     EXEC     DLIBATCH,                                       00142000
//             COND=((8,LE,INIT),(8,LE,READ.G)),                       00143000
//             PSB=8888888,                                            00144000
//             MBR=DSNMTV01                                            00145000
//G.IMS        DD       DISP=SHR,DSN=7777777                            00146000
//             DD       DISP=SHR,DSN=5555555                            00147000
```

Figure 302. Sample JCL for EKYUCCUD for the Direct Technique (Part 3 of 4)

```
//IEFRDER      DD       UNIT=SYSDA,SPACE=(CYL,(20,10))                  00148000
//DFSVSAMP     DD       *                                               00149000
 2048,10             /*   Change   */                                  00150000
 4096,10             /*     as     */                                  00151000
IOBF=(4096,10,Y,Y)   /*  Required  */                                  00152000
OPTIONS,INSERT=SEQ                                                     00153000
OPTIONS,DL/I=ON,LOCK=ON,DISP=ON,SCHD=ON                                00154000
//DDOTV02      DD       DISP=(,PASS),DSN=&&DDOTV02,                     00155000
//            DCB=(LRECL=4092,BLKSIZE=4096,RECFM=VB),                  00156000
//            SPACE=(TRK,1),UNIT=SYSDA                                 00157000
//DDITV02      DD       *                                               00158000
 2222222,,DSNMIN10,,R,,7777777,3333333,EKYC600X                        00159000
/*                                                                     00160000
//EKYRESLB     DD       DISP=SHR,                                       00161000
//            DSN=1111111.EKYRESLB                                     00162000
//CCUPRINT     DD       SYSOUT=*                                        00163000
//CCUCDS       DD       DISP=SHR,                                       00164000
//            DSN=1111111.CCUCDS                                       00165000
//CCUMSMTC     DD       DISP=OLD,                                       00166000
//            DSN=1111111.CCUMSMTC                                     00167000
//CCUIERR      DD       DISP=(,CATLG),                                  00168000
//            DSN=1111111.CCUIERR,                                     00169000
//            UNIT=SYSDA,SPACE=(CYL,(1,10),RLSE)                       00170000
//CCUDERR      DD       DISP=(,CATLG),                                  00171000
//            DSN=1111111.CCUDERR,                                     00172000
//            UNIT=SYSDA,SPACE=(CYL,(1,10),RLSE)                       00173000
//DATABASE     DD        DISP=SHR,DSN=???????                           00174000
//INDEX        DD        DISP=SHR,DSN=???????                           00175000
//*-----------------------------------------------------------------*/  00176000
//*                                                                 */  00177000
//*           SORT DB2 REPAIR STATEMENT FILE                        */  00178000
//*                                                                 */  00179000
//*-----------------------------------------------------------------*/  00180000
//SORT         EXEC     SORT,                                           00181000
//            COND=(8,LE,ERRORLOC.G)                                   00182000
//SORTWK01     DD       UNIT=SYSDA,SPACE=(CYL,(5,2))                    00183000
//SORTWK02     DD       UNIT=SYSDA,SPACE=(CYL,(5,2))                    00184000
//SORTWK03     DD       UNIT=SYSDA,SPACE=(CYL,(5,2))                    00185000
//SORTIN       DD       DISP=OLD,DCB=BUFNO=40,                          00186000
//            DSN=1111111.CCUDERR                                      00187000
//SORTOUT      DD       DISP=OLD,DCB=BUFNO=40,                          00188000
//            DSN=1111111.CCUDERR                                      00189000
//SYSIN        DD       *                                               00190000
  SORT FIELDS=(74,7,CH,A)                                              00191000
//SYSOUT       DD       SYSOUT=*                                        00192000
```

Figure 302. Sample JCL for EKYUCCUD for the Direct Technique (Part 4 of 4)

EKYUCCUD–Sample JCL for Direct Technique with the READONLY Keyword

Figure 302 on page 426 shows the JCL for the IMS/DB2 direct technique read phase without the READONLY keyword. Figure 303 on page 430 shows the JCL for the IMS/DB2 direct technique read phase when the READONLY keyword was used on the CHECK control statement in the initialization phase.

```
//USRT0022 JOB (H829415-TSOO),
//            USRT002,
//            USER=USRT002,PASSWORD=USRT002,
//            REGION=4096K,
//            NOTIFY=USRT002,CLASS=C,MSGCLASS=H
//*-----------------------------------------------------------------*/
//*                                                                 */
//*            R E A D   A N D   C O M P A R E   P H A S E          */
//*            F O R   D I R E C T   T E C H N I Q U E              */
//*-----------------------------------------------------------------*/
//SCRCCU       EXEC PGM=IEFBR14,COND=(04,LT)
//DCCUDERR     DD DSN=DPNRTCG.TT2A2055.CCUDERRH,
//             DISP=(MOD,DELETE),UNIT=3380,SPACE=(TRK,0)
//DCCUIERR     DD DSN=DPNRTCG.TT2A2055.CCUIERRH,
//             DISP=(MOD,DELETE),UNIT=3380,SPACE=(TRK,0)
//CCUMSMTC     DD DSN=DPNRTCG.TT2A2055.CCUMSMTH,
//             DISP=(MOD,DELETE),UNIT=3380,SPACE=(TRK,0)
//*-----------------------------------------------------------------*/
//ALC          EXEC PGM=IEBGENER,COND=(04,LT)
//SYSPRINT     DD SYSOUT=*
//SYSIN        DD DUMMY
//SYSUT1       DD DUMMY,DCB=(LRECL=80,BLKSIZE=7440,RECFM=FB)
//SYSUT2       DD DSN=DPNRTCG.TT2A2055.CCUDERRH,
//             DCB=(LRECL=80,BLKSIZE=7440,RECFM=FB),
//             UNIT=3380,SPACE=(TRK,(1,15)),DISP=(,CATLG)
//             DD DSN=DPNRTCG.TT2A2055.CCUIERRH,
//             DCB=(LRECL=80,BLKSIZE=7440,RECFM=FB),
//             UNIT=3380,SPACE=(TRK,(1,15)),DISP=(,CATLG)
//             DD DSN=DPNRTCG.TT2A2055.CCUMSMTH,
//             DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB),
//             UNIT=3380,SPACE=(TRK,(1,15)),DISP=(,CATLG)
//*-----------------------------------------------------------------*/
//T205501D     EXEC PGM=DFSRRC00,
//             PARM=(DLI,DSNMTV01,PSBH2,,,,,,,,,,,,,Y),
//             COND=(04,LT)
//IMS          DD DISP=SHR,DSN=DPNRTCG.TESTTEAM.V2R2.SJ.PSBLIB
//*            DD DISP=SHR,DSN=DPNRTCG.USRT002.LOAD
//             DD DISP=SHR,DSN=DPNRTCG.TESTTEAM.V2R2.SJ.DBDLIB
//RECON1       DD DISP=SHR,DSN=DPNRTCG.TT2A2055.RECON1
//RECON2       DD DISP=SHR,DSN=DPNRTCG.TT2A2055.RECON2
//RECON3       DD DISP=SHR,DSN=DPNRTCG.TT2A2055.RECON3
//IEFRDER      DD UNIT=SYSDA,SPACE=(TRK,(30,20),RLSE)
//DFSVSAMP     DD *
2048,8
4096,8
IOBF=(4096,6)
OPTIONS,DL/I=ON,LOCK=ON,DISP=ON,SCHD=ON
/*
//DDOTV02      DD SYSOUT=*,DCB=(RECFM=VB,LRECL=4092,BLKSIZE=4096)
//DDITV02      DD *
 DSH1,,DSNMIN10,,,,TT2A2055,UUSRT002,EKYC300X
/*
//SYSUDUMP     DD SYSOUT=*
//EKYTRACE     DD SYSOUT=*
//CCUPRINT     DD DISP=MOD,DSN=DPNRTCG.TT2A2055.EKYPRINT
//CCUCDS       DD DISP=SHR,DSN=DPNRTCG.TT2A2055.CCUCDSH
//CCUDERR      DD DISP=OLD,DSN=DPNRTCG.TT2A2055.CCUDERRH
//CCUIERR      DD DISP=OLD,DSN=DPNRTCG.TT2A2055.CCUIERRH
//CCUMSMTC     DD DISP=OLD,DSN=DPNRTCG.TT2A2055.CCUMSMTH
//*DFSRESLB    DD DISP=SHR,DSN=IMSESA.DPROP.510.RESLIB
//DFSRESLB     DD DISP=SHR,DSN=IMSBLD.I51STS51.CRESLIB
```

Figure 303. Sample JCL for CCU Read and Compare Phase (Part 1 of 2)

```
//* JSTEPLIB
//* STEPLIB FOR WHEN DATAREFRESHER IS REQ'D
//STEPLIB      DD DISP=SHR,
//             DSN=DPNRTCG.TESTTEAM.V2R2.SJ.LOAD         TCG STUFF
//    DD DISP=SHR,DSN=DPNR04.USRT002.EKYRESLB            MY GEND
//             DD DISP=SHR,DSN=DPNR.V2R2.A.EKYRESLB      2.2 LIBRARY
//             DD DISP=SHR,DSN=IMSBLD.I51STS51.CRESLIB   IMS
//             DD DISP=SHR,
//             DSN=DPNRTCG.TT2A2055.LOAD     ** SQL UPDATE MODULES
//             DD DISP=SHR,DSN=PAS110.DVRLOAD,  DATAREFRESHER
// UNIT=SYSALLDA,VOL=SER=DPR001
//             DD DISP=SHR,DSN=USER.TESTLIB     ZPARM
//             DD DISP=SHR,DSN=DSHC410.RUNLIB.LOAD
//             DD DISP=SHR,DSN=DSN410.SDSNLOAD
//* JRESLIB
//*EKYRESLB     DD DISP=SHR,DSN=EKY12.BASE.EKYRESLB
//EKYRESLB     DD DISP=SHR,DSN=DPNR04.USRT002.EKYRESLB
//             DD DISP=SHR,DSN=DPNR.V2R2.A.EKYRESLB
//TDBDPH1      DD DISP=SHR,DSN=DPNRTCG.TT2A2055.TDBDPH.$01
//TDBDPH2      DD DISP=SHR,DSN=DPNRTCG.TT2A2055.TDBDPH.$02
//T2DBDPD1     DD DISP=SHR,DSN=DPNRTCG.TT2A2055.T2DBDPD.$01
//T2DBDPD2     DD DISP=SHR,DSN=DPNRTCG.TT2A2055.T2DBDPD.$02
//T2DBDID      DD DISP=SHR,DSN=DPNRTCG.TT2A2055.T2DBDID.$01
//T2DBDPI      DD DISP=SHR,DSN=DPNRTCG.TT2A2055.T2DBDPI.$01
//T2DBDPO      DD DISP=SHR,DSN=DPNRTCG.TT2A2055.T2DBDPI.OFL.$01
```

Figure 303. Sample JCL for CCU Read and Compare Phase (Part 2 of 2)

EKYUCCUH–Sample JCL for the Hashing Technique

Figure 304 on page 432 is the sample JCL for EKYUCCUH, which uses a hashing technique for the read and compare phase of the CCU.

```
//JJJJJJJ JOB  ***  VALID JOB CARD DETAILS  ***                        00010000
//******************************************************************** 00020000
//**                                                                  * 00030000
//**        LICENSED MATERIALS - PROPERTY OF IBM                      * 00040000
//**                                                                  * 00050000
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                    * 00060000
//**                                                                  * 00070000
//**                                                                  * 00080000
//**        SEE COPYRIGHT INSTRUCTIONS                                * 00090000
//**                                                                  * 00100000
//******************************************************************** 00110000
//******************************************************************** 00130000
//*                                                                  */ 00140000
//*                 SAMPLE JOB STREAM FOR THE                        */ 00150000
//*              CONSISTENCY CHECK UTILITY (CCU)                     */ 00160000
//*                     HASHING TECHNIQUE                            */ 00170000
//*                                                                  */ 00180000
//*                                                                  */ 00190000
//*       THIS JOB CAN BE USED AS A SAMPLE JOB CONTROL STREAM        */ 00200000
//*       IF YOU HAVE NOT PLANNED TO USE THE DPROP CCU ISPF          */ 00210000
//*       SAMPLE APPLICATION.                                        */ 00220000
//*                                                                  */ 00230000
//*       THE FOLLOWING VARIABLES MUST BE ADAPTED IF YOU'RE NOT      */ 00240000
//*       USING THE DPROP INSTALLATION PROCEDURE:                    */ 00250000
//*                                                                  */ 00260000
//*         1111111   = DPROP HLQ(S)                                 */ 00270000
//*         2222222   = DB2 SUBSYSTEM ID                             */ 00280000
//*         3333333   = DB2 APPLICATION PLAN FOR CCU                */  00290000
//*                                                                 */  00300000
//*       THE FOLLOWING VALUES MUST BE CHANGED FOR YOUR ENVIRONMENT */  00310000
//*                                                                 */  00320000
//*         4444444    = IMS MACRO LIBRARY                          */  00330000
//*         5555555    = IMS DBD LOAD LIBRARY                       */  00340000
//*         6666666    = DBD NAME                                   */  00350000
//*         7777777    = IMS PSB LOAD LIBRARY                       */  00360000
//*         8888888    = NAME OF PSB GENERATED BY CCU               */  00370000
//*         VVVVVV     =VOLUME SERIAL NUMBER (WHERE APPROPRIATE)    */  00380000
//*                                                                 */  00390000
//*         //DATABASE - NAME OF THE IMS DATABASE TO BE CHECKED     */  00400000
//*         //INDEX    - NAME OF THE IMS DATABASE PRIMARY INDEX     */  00410000
//*                                                                 */  00420000
//*****************************************************************/  00430000
//*                                                                 */  00440000
//*----------------------------------------------------------------*/  00450000
//*                                                                 */  00460000
//*            DATA CONSISTENCY CHECK UTILITY                       */  00470000
//*                                                                 */  00480000
//*            HASHING TECHNIQUE IS BEING USED                      */  00490000
//*                                                                 */  00500000
```

Figure 304. Sample JCL for the EKYUCCUH procedure (Part 1 of 7)

```
//*-----------------------------------------------------------------*/  00510000
//*                                                                 */  00520000
//*           I N I T I A L I Z A T I O N   P H A S E               */  00530000
//*                                                                 */  00540000
//*-----------------------------------------------------------------*/  00550000
//DELETE       EXEC     PGM=IDCAMS                                      00560000
//SYSPRINT     DD       SYSOUT=*                                        00570000
//SYSIN        DD       *                                               00580000
 DELETE 1111111.CCUCDS                                                  00590000
//*--------------------*                                                00600000
//INIT        EXEC     PGM=IKJEFT01,DYNAMNBR=30                        00610000
//EKYRESLB     DD       DISP=SHR,                                       00620000
//             DSN=1111111.EKYRESLB                                     00630000
//SYSTSPRT     DD       SYSOUT=*                                        00640000
//SYSTSIN      DD       *                                               00650000
 DSN SYSTEM(2222222)                                                    00660000
 RUN PROGRAM(EKYC000X) -                                                00670000
     PLAN(3333333)                                                      00680000
//SYSPRINT     DD       SYSOUT=*                                        00690000
//SYSIN        DD       DUMMY                                           00700000
//ASYSIN       DD       DISP=(,PASS),DSN=&&ASYSIN,UNIT=SYSDA,           00710000
//             DCB=(BLKSIZE=720,RECFM=FB),SPACE=(TRK,1)                 00720000
//ASYSPRT      DD       SYSOUT=*                                        00730000
//ASYSLIB      DD       DISP=SHR,DSN=4444444.MACLIB                     00740000
//             DD       DISP=SHR,DSN=SYS1.MACLIB                        00750000
//ASYSLIN      DD       DISP=(,PASS),DSN=&&LKEDIN,UNIT=SYSDA,           00760000
//             DCB=(BLKSIZE=720,RECFM=FBS),SPACE=(CYL,1)                00770000
//LSYSPRT      DD       SYSOUT=*                                        00780000
//SYSLMOD      DD       DISP=SHR,DSN=7777777                            00790000
//SYSUT1       DD       UNIT=SYSDA,SPACE=(CYL,3)                        00800000
//SYSUT2       DD       UNIT=SYSDA,SPACE=(CYL,3)                        00810000
//SYSTERM      DD       SYSOUT=*                                        00820000
//CCUPRINT     DD       SYSOUT=*                                        00830000
//CCUCDS       DD       DISP=(,CATLG,DELETE),                           00840000
//             DSN=1111111.CCUCDS,                                      00850000
//             UNIT=SYSDA,SPACE=(TRK,(1,1),RLSE)                        00860000
//CCUDBD       DD       DISP=SHR,DSN=5555555                            00870000
//CCUIN        DD       *                                               00880000
 CHECK DBD=6666666                                                      00890000
     , PSBNAME=8888888                                                  00900000
 ;                                                                      00910000
```

Figure 304. Sample JCL for the EKYUCCUH procedure (Part 2 of 7)

```
//*-----------------------------------------------------------------*/  00920000
//*                                                                 */  00930000
//*           R E A D   A N D   C O M P A R E   P H A S E           */  00940000
//*                                                                 */  00950000
//*-----------------------------------------------------------------*/  00960000
//*           CCUKEY CLUSTER ALLOCATION                             */  00970000
//*-----------------------------------------------------------------*/  00980000
//ALCKEY1     EXEC     PGM=IDCAMS,COND=(8,LE,INIT)                      00990000
//SYSPRINT     DD       SYSOUT=*                                        01000000
//SYSIN        DD       *                                               01010000
    DELETE 1111111.CCUKEY1                                             01020000
       SET MAXCC = 0                                                   01030000
    DEFINE                                  -                          01040000
       CLUSTER (NAME (1111111.CCUKEY1) -                               01050000
                BUFFERSPACE   (32768)        -                         01060000
                CISZ          (32768)        -                         01070000
                NOERASE                      -                         01080000
                INDEXED                      -                         01090000
                NOIMBED                      -                         01100000
                KEYS            (4 0)        -                         01110000
                RECORDSIZE (50 32761)        -                         01120000
                NOREPLICATE                  -                         01130000
                NOREUSE                      -                         01140000
                SHAREOPTIONS    (3 3)        -                         01150000
                SPEED                        -                         01160000
                CYLINDERS       (1 5)        -                         01170000
                VOLUMES      (VVVVVV) )      -                         01180000
       INDEX   (NAME (1111111.CCUKEY1.INDEX) ) -                       01190000
       DATA    (NAME (1111111.CCUKEY1.DATA) -                          01200000
                RECORDSIZE (50 32761) )                                01210000
                                                                       01220000
    DELETE 1111111.CCUKEY2                                             01230000
       SET MAXCC = 0                                                   01240000
    DEFINE                                  -                          01250000
       CLUSTER (NAME (1111111.CCUKEY2) -                               01260000
                BUFFERSPACE   (32768)        -                         01270000
                CISZ          (32768)        -                         01280000
                NOERASE                      -                         01290000
                INDEXED                      -                         01300000
                NOIMBED                      -                         01310000
                KEYS            (4 0)        -                         01320000
                RECORDSIZE (50 32761)        -                         01330000
                NOREPLICATE                  -                         01340000
                NOREUSE                      -                         01350000
                SHAREOPTIONS    (3 3)        -                         01360000
                SPEED                        -                         01370000
                CYLINDERS       (1 5)        -                         01380000
                VOLUMES      (VVVVVV) )      -                         01390000
       INDEX   (NAME (1111111.CCUKEY2.INDEX) ) -                       01400000
       DATA    (NAME (1111111.CCUKEY2.DATA) -                          01410000
                RECORDSIZE (50 32761) )                                01420000
```

Figure 304. Sample JCL for the EKYUCCUH procedure (Part 3 of 7)

```
//*-----------------------------------------------------------------*/  01430000
//*            IMS READ STEP                                         */  01440000
//*-----------------------------------------------------------------*/  01450000
//DELETE       EXEC     PGM=IDCAMS,COND=(8,LE,INIT)                     01460000
//SYSPRINT     DD       SYSOUT=*                                        01470000
//SYSIN        DD       *                                               01480000
 DELETE '1111111.CCUHASH1'                                             01490000
 DELETE '1111111.CCUHSUM1'                                             01500000
//*--------------------*                                               01510000
//IMSREAD      EXEC     DLIBATCH,COND=(8,LE,INIT),                      01520000
//             MBR=DSNMTV01,         * DB2 DRIVER MODULE NAME *        01530000
//             PSB=7777777,          * DL/I PSBNAME           *        01540000
//             SOUT='*'              * SYSOUT CLASS           *        01550000
//*                                                                    01560000
//G.IMS        DD       DISP=SHR,DSN=7777777                            01570000
//             DD       DISP=SHR,DSN=5555555                            01580000
//IEFRDER      DD       UNIT=SYSDA,SPACE=(CYL,(20,10))                  01590000
//SYSUDUMP     DD       SYSOUT=*                                        01600000
//DFSVSAMP     DD       *                                               01610000
1024,6             /*   Change     */                                  01620000
2048,6             /* these values */                                  01630000
4096,6             /*     as       */                                  01640000
IOBF=(4096,6,N,N)  /*   required   */                                  01650000
OPTIONS,DL/I=ON,LOCK=ON,DISP=ON,SCHD=ON                                01660000
//EKYRESLB     DD       DISP=SHR,                                       01670000
//             DSN=1111111.EKYRESLB                                    01680000
//DDOTV02      DD       DISP=(,PASS),DSN=&&DDITV02,                     01690000
//             DCB=(LRECL=4092,BLKSIZE=4096,RECFM=VB),                 01700000
//             SPACE=(TRK,1),UNIT=SYSDA                                01710000
//DDITV02      DD       *                                               01720000
 2222222,,DSNMIN10,,R,,CCUDLIRD,3333333,EKYC100X                       01730000
/*                                                                     01740000
//CCUPRINT     DD       SYSOUT=*                                        01750000
//CCUCDS       DD       DISP=SHR,                                       01760000
//             DSN=1111111.CCUCDS                                      01770000
//CCUHSUM1     DD       DISP=(,CATLG),                                  01780000
//             DSN=1111111.CCUHSUM1,                                   01790000
//             DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB),               01800000
//             UNIT=SYSDA,SPACE=(TRK,1)                                01810000
//CCUHASH1     DD       DISP=(,CATLG),                                  01820000
//             DSN=1111111.CCUHASH1,                                   01830000
//             DCB=(LRECL=16,BLKSIZE=7472,RECFM=FB,BUFNO=40),          01840000
//             UNIT=SYSDA,SPACE=(CYL,(1,10),RLSE)                      01850000
//CCUKEY1      DD       DISP=OLD,                                       01860000
//             DSN=1111111.CCUKEY1                                     01870000
//*                                                                    01880000
//DATABASE     DD       DISP=SHR,DSN=???????                            01890000
//INDEX        DD       DISP=SHR,DSN=???????                            01900000
//*--------------------*                                               01910000
//PRINT        EXEC     PGM=DFSERA10,COND=((8,LE,INIT),(EVEN))          01920000
//SYSPRINT     DD       SYSOUT=*                                        01930000
//SYSUT1       DD       DISP=(OLD,DELETE),DSN=&&DDITV02                 01940000
//SYSIN        DD       *                                               01950000
CONTROL  CNTL K=000,H=8000                                             01960000
OPTION   PRINT                                                         01970000
```

Figure 304. Sample JCL for the EKYUCCUH procedure (Part 4 of 7)

```
//*-----------------------------------------------------------------*/  01980000
//*            DB2 READ STEP                                        */  01990000
//*-----------------------------------------------------------------*/  02000000
//DELETE       EXEC     PGM=IDCAMS,COND=(8,LE,INIT)                     02010000
//SYSPRINT     DD       SYSOUT=*                                        02020000
//SYSIN        DD       *                                               02030000
 DELETE '1111111.CCUHASH2'                                              02040000
 DELETE '1111111.CCUHSUM2'                                              02050000
//*--------------------*                                                02060000
//DB2READ     EXEC     PGM=IKJEFT01,DYNAMNBR=30,COND=(8,LE,INIT)       02070000
//EKYRESLB     DD       DISP=SHR,                                       02080000
//             DSN=1111111.EKYRESLIB                                    02090000
//SYSPRINT     DD       SYSOUT=*                                        02100000
//SYSTSPRT     DD       SYSOUT=*                                        02110000
//SYSTSIN      DD       *                                               02120000
 DSN SYSTEM(2222222)                                                    02130000
 RUN PROGRAM(EKYC200X) -                                                02140000
     PLAN(3333333)                                                      02150000
//SYSIN        DD       DUMMY                                           02160000
//CCUPRINT     DD       SYSOUT=*                                        02170000
//CCUCDS       DD       DISP=SHR,                                       02180000
//             DSN=1111111.CCUCDS                                       02190000
//CCUHSUM2     DD       DISP=(,CATLG),                                  02200000
//             DSN=1111111.CCUHSUM2,                                    02210000
//             DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB),                02220000
//             UNIT=SYSDA,SPACE=(TRK,1)                                 02230000
//CCUHASH2     DD       DISP=(,CATLG),                                  02240000
//             DSN=1111111.CCUHASH2,                                    02250000
//             DCB=(LRECL=16,BLKSIZE=7472,RECFM=FB,BUFNO=40),           02260000
//             UNIT=SYSDA,SPACE=(CYL,(1,10),RLSE)                       02270000
//CCUKEY2      DD       DISP=OLD,                                       02280000
//             DSN=1111111.CCUKEY2                                      02290000
//*-----------------------------------------------------------------*/  02300000
//*                                                                 */  02310000
//*            HASH SUM COMPARE STEP                                */  02320000
//*                                                                 */  02330000
//*-----------------------------------------------------------------*/  02340000
//DELETE       EXEC     PGM=IDCAMS,                                     02350000
//             COND=((8,LE,INIT),(8,LE,IMSREAD.G),(8,LE,DB2READ))      02360000
//SYSPRINT     DD       SYSOUT=*                                        02370000
//SYSIN        DD       *                                               02380000
 DELETE '1111111.CCUSORTS'                                              02390000
//*--------------------*                                                02400000
//HSUMCOMP    EXEC     PGM=EKYC400X,                                    02410000
//             COND=((8,LE,INIT),(8,LE,IMSREAD.G),(8,LE,DB2READ))      02420000
//EKYRESLB     DD       DISP=SHR,                                       02430000
//             DSN=1111111.EKYRESLB                                     02440000
//SYSPRINT     DD       SYSOUT=*                                        02450000
//SYSUDUMP     DD       SYSOUT=*                                        02460000
//CCUPRINT     DD       SYSOUT=*                                        02470000
//CCUCDS       DD       DISP=SHR,                                       02480000
//             DSN=1111111.CCUCDS                                       02490000
//CCUHSUM1     DD       DISP=OLD,                                       02500000
//             DSN=1111111.CCUHSUM1                                     02510000
//CCUHSUM2     DD       DISP=OLD,                                       02520000
//             DSN=1111111.CCUHSUM2                                     02530000
//CCUSORTS     DD       DISP=(,CATLG),                                  02540000
//             DSN=1111111.CCUSORTS,                                    02550000
//             DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),                     02560000
//             UNIT=SYSDA,SPACE=(TRK,1)                                 02570000
```

Figure 304. Sample JCL for the EKYUCCUH procedure (Part 5 of 7)

```
//*-----------------------------------------------------------------*/  02580000
//*                                                                 */  02590000
//*            COMPARE STEP                                         */  02600000
//*                                                                 */  02610000
//*-----------------------------------------------------------------*/  02620000
//DELETE       EXEC     PGM=IDCAMS,                                     02630000
// COND=((8,LE,INIT),(8,LE,IMSREAD.G),(8,LE,DB2READ),(4,NE,HSUMCOMP))   02640000
//SYSPRINT     DD       SYSOUT=*                                        02650000
//SYSIN        DD       *                                               02660000
 DELETE '1111111.CCUMSMTC'                                              02670000
//*--------------------*                                                02680000
//COMPARE     EXEC     PGM=EKYC500X,REGION=0M,                          02690000
// COND=((8,LE,INIT),(8,LE,IMSREAD.G),(8,LE,DB2READ),(4,NE,HSUMCOMP))   02700000
//EKYRESLB     DD       DISP=SHR,                                       02710000
//             DSN=1111111.EKYRESLB                                     02720000
//SORTWK01     DD       UNIT=SYSDA,SPACE=(CYL,(5,2))                    02730000
//SORTWK02     DD       UNIT=SYSDA,SPACE=(CYL,(5,2))                    02740000
//SORTWK03     DD       UNIT=SYSDA,SPACE=(CYL,(5,2))                    02750000
//SORTCNTL     DD       DISP=SHR,                                       02760000
//             DSN=1111111.CCUSORTS                                     02770000
//SYSOUT       DD       SYSOUT=*                                        02780000
//SYSIN        DD       DISP=OLD,                                       02790000
//             DSN=1111111.CCUSORTS                                     02800000
//SYSUDUMP     DD       SYSOUT=*                                        02810000
//CCUPRINT     DD       SYSOUT=*                                        02820000
//CCUCDS       DD       DISP=SHR,                                       02830000
//             DSN=1111111.CCUCDS                                       02840000
//CCUSORTI     DD       DISP=OLD,DCB=BUFNO=40,                          02850000
//             DSN=1111111.CCUHASH1                                     02860000
//             DD       D1SP=OLD,DCB=BUFNO=40,                          02870000
//             DSN=1111111.CCUHASH2                                     02880000
//CCUMSMTC     DD       DISP=(,CATLG),                                  02890000
//             DSN=1111111.CCUMSMTC,                                    02900000
//             DCB=(LRECL=16,BLKSIZE=7472,RECFM=FB,BUFNO=40),           02910000
//             UNIT=SYSDA,SPACE=(CYL,(1,10),RLSE)                       02920000
//*-----------------------------------------------------------------*/  02930000
//*                                                                 */  02940000
//*            E R R O R    L O C A T I O N    P H A S E             */  02950000
//*                                                                 */  02960000
//*-----------------------------------------------------------------*/  02970000
//DELETE       EXEC     PGM=IDCAMS,COND=(8,LE,INIT)                     02980000
//SYSPRINT     DD       SYSOUT=*                                        02990000
//SYSIN        DD       *                                               03000000
 DELETE '1111111.CCUDERR'                                               03010000
 DELETE '1111111.CCUIERR'                                               03020000
//*--------------------*                                                03030000
//ERRORLOC     EXEC     DLIBATCH,                                       03040000
//             COND=((8,LE,INIT),(8,LE,IMSREAD.G),(8,LE,DB2READ),      03050000
//             (4,NE,HSUMCOMP),(4,LE,COMPARE)),                         03060000
//             PSB=7777777,MBR=DSNMTV01                                 03070000
//G.IMS        DD       DISP=SHR,DSN=7777777                            03080000
//             DD       DISP=SHR,DSN=5555555                            03090000
//IEFRDER      DD       UNIT=SYSDA,SPACE=(CYL,(20,10))                  03100000
//DFSVSAMP     DD       *                                               03110000
 2048,10            /*   Change     */                                 03120000
 4096,10            /* these value  */                                 03130000
IOBF=(4096,10,Y,Y)  /* as required  */                                 03140000
OPTIONS,INSERT=SEQ                                                      03150000
OPTIONS,DL/I=ON,LOCK=ON,DISP=ON,SCHD=ON                                 03160000
```

Figure 304. Sample JCL for the EKYUCCUH procedure (Part 6 of 7)

```
//DDOTV02      DD       DISP=(,PASS),DSN=&&DDOTV02,                    03170000
//             DCB=(LRECL=4092,BLKSIZE=4096,RECFM=VB),                 03180000
//             SPACE=(TRK,1),UNIT=SYSDA                                03190000
//DDITV02      DD       *                                              03200000
 2222222,,DSNMIN10,,R,,CCUERRLC,3333333,EKYC600X                       03210000
/*                                                                     03220000
//EKYRESLB     DD       DISP=SHR,                                      03230000
//             DSN=1111111.EKYRESLB                                    03240000
//CCUPRINT     DD       SYSOUT=*                                       03250000
//CCUCDS       DD       DISP=SHR,                                      03260000
//             DSN=1111111.CCUCDS                                      03270000
//CCUMSMTC     DD       DISP=OLD,                                      03280000
//             DSN=1111111.CCUMSMTC                                    03290000
//CCUDERR      DD       DISP=(,CATLG),                                 03300000
//             DSN=1111111.CCUDERR,                                    03310000
//             UNIT=SYSDA,SPACE=(CYL,(1,10),RLSE)                      03320000
//CCUIERR      DD       DISP=(,CATLG),                                 03330000
//             DSN=1111111.CCUIERR,                                    03340000
//             UNIT=SYSDA,SPACE=(CYL,(1,10),RLSE)                      03350000
//CCUKEY1      DD       DISP=OLD,                                      03360000
//             DSN=1111111.CCUKEY1                                     03370000
//CCUKEY2      DD       DISP=OLD,                                      03380000
//             DSN=1111111.CCUKEY2                                     03390000
//DATABASE     DD        DISP=SHR,DSN=???????                          03400000
//INDEX        DD        DISP=SHR,DSN=???????                          03410000
```

Figure 304. Sample JCL for the EKYUCCUH procedure (Part 7 of 7)

EKYUCCHD–Sample JCL for the Hashing Technique with an HD Unload file

EKYUCCHD executes the CCU using the hashing technique. The input to the read and compare phase is an HD unload file instead of the IMS database.

```
//JJJJJJJ JOB  ***  VALID JOB CARD DETAILS  ***                          00010000
//******************************************************************** 00020000
//**                                                                  * 00030000
//**        LICENSED MATERIALS - PROPERTY OF IBM                      * 00040000
//**        5696-705 (C) COPYRIGHT IBM CORP. 1994.                    * 00060000
//**        SEE COPYRIGHT INSTRUCTIONS                                * 00090000
//**                                                                  * 00100000
//******************************************************************** 00110000
//******************************************************************** 00130000
//*                                                                 */ 00140000
//*               SAMPLE JOB STREAM FOR THE                         */ 00150000
//*              CONSISTENCY CHECK UTILITY (CCU)                    */ 00160000
//*              HASHING TECHNIQUE USING AN                         */ 00170000
//*              IMS HDUNLOAD FILE AS INPUT                         */ 00180000
//*                                                                 */ 00190000
//*       THIS JOB CAN BE USED AS A SAMPLE JOB CONTROL STREAM       */ 00200000
//*       IF YOU HAVE NOT PLANNED TO USE THE DPROP CCU ISPF         */ 00210000
//*       SAMPLE APPLICATION.                                       */ 00220000
//*                                                                 */ 00230000
//*       THE FOLLOWING VARIABLES MUST BE ADAPTED                   */ 00240000
//*                                                                 */ 00250000
//*         1111111   = DPROP HLQ(S)                                */ 00260000
//*         2222222   = DB2 SUBSYSTEM ID                            */ 00270000
//*         3333333   = DB2 APPLICATION PLAN FOR CCU                */ 00280000
//*                                                                 */ 00290000
//*         4444444    = IMS DBD LOAD LIBRARY                       */ 00300000
//*         5555555    = DBD NAME                                   */ 00310000
//*         6666666    = HDUNLOAD FILE DSN                         */  00320000
//*                                                                */  00330000
//*****************************************************************/  00340000
//*----------------------------------------------------------------*/  00360000
//*                                                                */  00370000
//*           DATA CONSISTENCY CHECK UTILITY                       */  00380000
//*                                                                */  00390000
//*           HASHING TECHNIQUE IS BEING USED                      */  00400000
//*                                                                */  00410000
//*----------------------------------------------------------------*/  00420000
//*----------------------------------------------------------------*/  00430000
//DELETE      EXEC     PGM=IDCAMS                                     00440000
//SYSPRINT     DD       SYSOUT=*                                      00450000
//SYSIN        DD       *                                             00460000
 DELETE '1111111.CCUCDS'                                              00470000
 IF MAXCC = 8 THEN SET MAXCC = 0                                      00480000
//*----------------------------------------------------------------*/  00490000
//*                                                                */  00500000
//*           I N I T I A L I Z A T I O N   P H A S E              */  00510000
//*                                                                */  00520000
//*----------------------------------------------------------------*/  00530000
//INIT        EXEC     PGM=IKJEFT01,DYNAMNBR=30,COND=(4,LT)           00540000
//SYSTSIN      DD       *                                             00550000
 DSN SYSTEM(2222222)                                                  00560000
 RUN PROGRAM(EKYC000X) PLAN(3333333)                                  00570000
//SYSIN        DD       DUMMY                                         00580000
//SYSPRINT     DD       SYSOUT=*                                      00590000
//SYSTSPRT     DD       SYSOUT=*                                      00600000
//SYSTERM      DD       SYSOUT=*                                      00610000
//EKYRESLB     DD       DISP=SHR,                                     00620000
//            DSN=1111111.EKYRESLB                                    00630000
//CCUPRINT     DD       SYSOUT=*                                      00640000
//CCUCDS       DD       DISP=(,CATLG,DELETE),                         00650000
```

Figure 305. Sample JCL for EKYUCCHD (Part 1 of 6)

```
//             SPACE=(TRK,(5,5),RLSE),                                  00660000
//             DSN=1111111.CCUCDS                                       00670000
//CCUDBD       DD       DISP=SHR,                                       00680000
//             DSN=4444444                                              00690000
//CCUIN        DD       *                                               00700000
 CHECK DBD=5555555                                                     00710000
    , MAXERROR=200                                                     00720000
 ;                                                                     00730000
//EKYTRACE     DD       SYSOUT=*                                        00740000
//*-----------------------------------------------------------------*/  00750000
//*            R E A D    A N D    C O M P A R E    P H A S E      */  00770000
//*-----------------------------------------------------------------*/  00790000
//*    DUMMY DELETE CCUKEY FILES FOR JES3 SATISFACTION               * 00800000
//*-----------------------------------------------------------------* 00810000
//DMYDEL1   EXEC     PGM=IDCAMS                                        00820000
//SYSPRINT     DD       SYSOUT=*                                       00830000
//SYSIN        DD       *                                              00840000
 DELETE 1111111.CCUKEY1                                                00850000
 SET MAXCC = 0                                                         00860000
//*-----------------------------------------------------------------* 00870000
//*    DUMMY ALLOC CCUKEY FILES FOR JES3 SATISFACTION                * 00880000
//*-----------------------------------------------------------------* 00890000
//DMYALC1   EXEC     PGM=IEFBR14                                       00900000
//DD1       DD       DSN=1111111.CCUKEY1,                              00910000
//             SPACE=(TRK,1),DISP=(,CATLG),                             00920000
//             DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),                     00930000
//             UNIT=SYSDA                                               00940000
//*-----------------------------------------------------------------*/  00950000
//*            CCUKEY1 CLUSTER ALLOCATION                           */  00960000
//*-----------------------------------------------------------------*/  00970000
//ALCKEY1     EXEC     PGM=IDCAMS,COND=(4,LT)                           00980000
//SYSPRINT     DD       SYSOUT=*                                       00990000
//SYSIN        DD       *                                              01000000
    DELETE 1111111.CCUKEY1                                             01010000
       IF MAXCC = 8 THEN SET MAXCC = 0                                 01020000
    DEFINE -                                                           01030000
       CLUSTER (NAME (1111111.CCUKEY1) -                               01040000
                BUFFERSPACE   (32768) -                                01050000
                CISZ          (32768) -                                01060000
                NOERASE               -                                01070000
                INDEXED               -                                01080000
                NOIMBED               -                                01090000
                KEYS            (4 0) -                                01100000
                RECORDSIZE (50 32761) -                                01110000
                NOREPLICATE           -                                01120000
                NOREUSE               -                                01130000
                SHAREOPTIONS    (3 3) -                                01140000
                TRK    (10,10)   -                                     01150000
                SPEED                 ) -                              01160000
       INDEX   (NAME (1111111.CCUKEY1.INDEX) ) -                       01170000
       DATA    (NAME (1111111.CCUKEY1.DATA) -                          01180000
                RECORDSIZE (50 32761) )                                01190000
//*-----------------------------------------------------------------*/  01200000
//*            IMS READ STEP                                        */  01210000
//*                IMS DATA INPUT IS ON HD UNLOAD FILE              */  01230000
//*-----------------------------------------------------------------*/  01240000
//DELIMSR     EXEC     PGM=IDCAMS,COND=(4,LT)                           01250000
//SYSPRINT     DD       SYSOUT=*                                       01260000
//SYSIN        DD       *                                              01270000
 DELETE '1111111.CCUHSUM1'                                             01280000
 DELETE '1111111.CCUHASH1'                                             01290000
 DELETE '1111111.CCUDBOUT'                                             01300000
 IF MAXCC = 8 THEN SET MAXCC = 0                                       01310000
```

Figure 305. Sample JCL for EKYUCCHD (Part 2 of 6)

```
//*-----------------------------------------------------------------*/  01320000
//IMSREAD     EXEC     PGM=IKJEFT01,DYNAMNBR=30,COND=(4,LT)             01330000
//SYSTSPRT     DD       SYSOUT=*                                       01340000
//SYSTSIN      DD       *                                              01350000
 DSN SYSTEM(2222222)                                                   01360000
 RUN PROGRAM(EKYC100X) PLAN(3333333)                                   01370000
//SYSPRINT     DD       SYSOUT=*                                       01380000
//SYSUDUMP     DD       SYSOUT=*                                       01390000
//EKYRESLB     DD       DISP=SHR,                                      01400000
//             DSN=1111111.EKYRESLB                                    01410000
//CCUPRINT     DD       SYSOUT=*                                       01420000
//CCUCDS       DD       DISP=SHR,                                      01430000
//             DSN=1111111.CCUCDS                                      01440000
//CCUHSUM1     DD       DISP=(,CATLG),                                 01450000
//             SPACE=(TRK,(5,5),RLSE),                                 01460000
//             DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB),               01470000
//             DSN=1111111.CCUHSUM1                                    01480000
//CCUHASH1     DD       DISP=(,CATLG),                                 01490000
//             SPACE=(TRK,(10,10),RLSE),                               01500000
//             DCB=(LRECL=16,BLKSIZE=7472,RECFM=FB,BUFNO=40),          01510000
//             DSN=1111111.CCUHASH1                                    01520000
//CCUKEY1      DD       DISP=OLD,                                      01530000
//             DSN=1111111.CCUKEY1                                     01540000
//EKYTRACE     DD       SYSOUT=*                                       01550000
//CCUDBOUT     DD       DISP=(,CATLG),                                 01560000
//             SPACE=(TRK,(10,10),RLSE),                               01570000
//             DCB=BUFNO=40,                                           01580000
//             DSN=1111111.CCUDBOUT                                    01590000
//CCUDBIN      DD       DISP=SHR,                                      01600000
//             DSN=6666666                                             01610000
//*-----------------------------------------------------------------*/  01620000
//*    DUMMY DELETE CCUKEY FILES FOR JES3 SATISFACTION               * 01630000
//*-----------------------------------------------------------------* 01640000
//DMYDEL2   EXEC     PGM=IDCAMS                                       01650000
//SYSPRINT     DD       SYSOUT=*                                       01660000
//SYSIN        DD       *                                              01670000
 DELETE 1111111.CCUKEY2                                                01680000
 SET MAXCC = 0                                                         01690000
//*-----------------------------------------------------------------* 01700000
//*    DUMMY ALLOC CCUKEY FILES FOR JES3 SATISFACTION                * 01710000
//*-----------------------------------------------------------------* 01720000
//DMYALC2   EXEC     PGM=IEFBR14                                      01730000
//DD2       DD       DSN=1111111.CCUKEY2,                             01740000
//             SPACE=(TRK,1),DISP=(,CATLG),                            01750000
//             DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),                    01760000
//             UNIT=SYSDA                                              01770000
//*-----------------------------------------------------------------*/  01780000
//*            CCUKEY2 CLUSTER ALLOCATION                          */  01790000
//ALCKEY2     EXEC     PGM=IDCAMS,COND=(4,LT)                          01810000
//SYSPRINT     DD       SYSOUT=*                                       01820000
//SYSIN        DD       *                                              01830000
    DELETE 1111111.CCUKEY2                                             01840000
       IF MAXCC = 8 THEN SET MAXCC = 0                                 01850000
    DEFINE -                                                           01860000
       CLUSTER (NAME (1111111.CCUKEY2) -                               01870000
                BUFFERSPACE   (32768) -                                01880000
                CISZ          (32768) -                                01890000
                NOERASE               -                                01900000
                INDEXED               -                                01910000
                NOIMBED               -                                01920000
```

Figure 305. Sample JCL for EKYUCCHD (Part 3 of 6)

```
               KEYS           (4 0) -                                  01930000
               RECORDSIZE (50 32761) -                                 01940000
               NOREPLICATE           -                                 01950000
               NOREUSE               -                                 01960000
               SHAREOPTIONS    (3 3) -                                 01970000
               TRK    (10,10)   -                                      01980000
               SPEED                 ) -                               01990000
       INDEX   (NAME (1111111.CCUKEY2.INDEX) ) -                       02000000
       DATA    (NAME (1111111.CCUKEY2.DATA) -                          02010000
               RECORDSIZE (50 32761) )                                 02020000
//*-----------------------------------------------------------------*/ 02030000
//*           DB2 READ STEP                                         */ 02040000
//*-----------------------------------------------------------------*/ 02050000
//DELDB2R     EXEC     PGM=IDCAMS,COND=(4,LT)                         02060000
//SYSPRINT     DD       SYSOUT=*                                      02070000
//SYSIN        DD       *                                             02080000
 DELETE '1111111.CCUHSUM2'                                             02090000
 DELETE '1111111.CCUHASH2'                                             02100000
 IF MAXCC = 8 THEN SET MAXCC = 0                                       02110000
//*-----------------------------------------------------------------*/ 02120000
//DB2READ     EXEC     PGM=IKJEFT01,DYNAMNBR=30,COND=(4,LT)           02130000
//SYSPRINT     DD       SYSOUT=*                                      02140000
//SYSTSPRT     DD       SYSOUT=*                                      02150000
//SYSTSIN      DD       *                                             02160000
 DSN SYSTEM(2222222)                                                   02170000
 RUN PROGRAM(EKYC200X) PLAN(3333333)                                   02180000
//SYSUDUMP     DD       SYSOUT=*                                      02190000
//EKYRESLB     DD       DISP=SHR,                                     02200000
//            DSN=1111111.EKYRESLB                                    02210000
//CCUPRINT     DD       SYSOUT=*                                      02220000
//CCUCDS       DD       DISP=SHR,                                     02230000
//            DSN=1111111.CCUCDS                                      02240000
//CCUHSUM2     DD       DISP=(,CATLG),                                02250000
//            SPACE=(TRK,(5,5),RLSE),                                 02260000
//            DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB),               02270000
//            DSN=1111111.CCUHSUM2                                    02280000
//CCUHASH2     DD       DISP=(,CATLG),                                02290000
//            SPACE=(TRK,(10,10),RLSE),                               02300000
//            DCB=(LRECL=16,BLKSIZE=7472,RECFM=FB,BUFNO=40),          02310000
//            DSN=1111111.CCUHASH2                                    02320000
//CCUKEY2      DD       DISP=OLD,                                     02330000
//            DSN=1111111.CCUKEY2                                     02340000
//EKYTRACE     DD       SYSOUT=*                                      02350000
//*-----------------------------------------------------------------*/ 02360000
//*           HASH SUM COMPARE STEP                                 */ 02380000
//*-----------------------------------------------------------------*/ 02400000
//DELHSUM     EXEC     PGM=IDCAMS,COND=(4,LT)                         02410000
//SYSPRINT     DD       SYSOUT=*                                      02420000
//SYSIN        DD       *                                             02430000
 DELETE '1111111.CCUSORTS'                                             02440000
 IF MAXCC = 8 THEN SET MAXCC = 0                                       02450000
//*-----------------------------------------------------------------*/ 02460000
//HSUMCOMP    EXEC     PGM=EKYC400X,COND=(4,LT)                       02470000
//SYSPRINT     DD       SYSOUT=*                                      02480000
//SYSUDUMP     DD       SYSOUT=*                                      02490000
//EKYRESLB     DD       DISP=SHR,                                     02500000
//            DSN=1111111.EKYRESLB                                    02510000
//CCUPRINT     DD       SYSOUT=*                                      02520000
//CCUCDS       DD       DISP=SHR,                                     02530000
//            DSN=1111111.CCUCDS                                      02540000
```

Figure 305. Sample JCL for EKYUCCHD (Part 4 of 6)

```
//CCUHSUM1     DD       DISP=OLD,                                        02550000
//             DSN=1111111.CCUHSUM1                                      02560000
//CCUHSUM2     DD       DISP=OLD,                                        02570000
//             DSN=1111111.CCUHSUM2                                      02580000
//CCUSORTS     DD       DISP=(,CATLG),                                   02590000
//             SPACE=(TRK,(5,5),RLSE),                                   02600000
//             DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),                      02610000
//             DSN=1111111.CCUSORTS                                      02620000
//EKYTRACE     DD       SYSOUT=*                                         02630000
//*-----------------------------------------------------------------*/  02640000
//*            COMPARE STEP                                          */  02660000
//*-----------------------------------------------------------------*/  02680000
//DELCOMP     EXEC     PGM=IDCAMS,COND=((4,LT),(4,NE,HSUMCOMP))         02690000
//SYSPRINT     DD       SYSOUT=*                                         02700000
//SYSIN        DD       *                                                02710000
 DELETE '1111111.CCUMSMTC'                                               02720000
 IF MAXCC = 8 THEN SET MAXCC = 0                                         02730000
//*-----------------------------------------------------------------*/  02740000
//COMPARE     EXEC     PGM=EKYC500X,REGION=0M,                          02750000
//             COND=((4,LT),(4,NE,HSUMCOMP))                            02760000
//SORTWK01     DD       UNIT=SYSDA,SPACE=(CYL,(5,10))                    02770000
//SORTWK02     DD       UNIT=SYSDA,SPACE=(CYL,(5,10))                    02780000
//SORTWK03     DD       UNIT=SYSDA,SPACE=(CYL,(5,10))                    02790000
//SORTCNTL     DD       DISP=SHR,                                        02800000
//             DSN=1111111.CCUSORTS                                      02810000
//SYSOUT       DD       SYSOUT=*                                         02820000
//SYSIN        DD       DISP=SHR,                                        02830000
//             DSN=1111111.CCUSORTS                                      02840000
//SYSUDUMP     DD       SYSOUT=*                                         02850000
//EKYRESLB     DD       DISP=SHR,                                        02860000
//             DSN=1111111.EKYRESLB                                      02870000
//CCUPRINT     DD       SYSOUT=*                                         02880000
//CCUCDS       DD       DISP=SHR,                                        02890000
//             DSN=1111111.CCUCDS                                        02900000
//CCUSORTI     DD       DISP=OLD,DCB=BUFNO=40,                           02910000
//             DSN=1111111.CCUHASH1                                      02920000
//             DD       DISP=OLD,DCB=BUFNO=40,                           02930000
//             DSN=1111111.CCUHASH2                                      02940000
//CCUMSMTC     DD       DISP=(,CATLG),                                   02950000
//             SPACE=(TRK,(5,5),RLSE),                                   02960000
//             DCB=(LRECL=16,BLKSIZE=7472,RECFM=FB,BUFNO=40),           02970000
//             DSN=1111111.CCUMSMTC                                      02980000
//CCUKEY1      DD       DISP=SHR,                                        02990000
//             DSN=1111111.CCUKEY1                                       03000000
//CCUKEY2      DD       DISP=SHR,                                        03010000
//             DSN=1111111.CCUKEY2                                       03020000
//EKYTRACE     DD       SYSOUT=*                                         03030000
//*-----------------------------------------------------------------*/  03040000
//*                                                                 */  03050000
//*            E R R O R    L O C A T I O N    P H A S E             */  03060000
//*                IMS DATA INPUT IS ON //CCUDBOUT DATA SET          */  03080000
//*                ORIGINATES FROM THE HD UNLOAD FILE                */  03090000
//*-----------------------------------------------------------------*/  03120000
//DELERRL     EXEC     PGM=IDCAMS,                                      03130000
//             COND=((4,LT),(4,NE,HSUMCOMP),(0,NE,COMPARE))             03140000
//SYSPRINT     DD       SYSOUT=*                                         03150000
//SYSIN        DD       *                                                03160000
 DELETE '1111111.CCUDERR'                                                03170000
 IF MAXCC = 8 THEN SET MAXCC = 0                                         03180000
```

Figure 305. Sample JCL for EKYUCCHD (Part 5 of 6)

```
//*-----------------------------------------------------------------*/  03190000
//RESORT      EXEC     SORT,                                            03200000
//             COND=((4,LT),(4,NE,HSUMCOMP),(0,NE,COMPARE))             03210000
//SORTIN       DD       DISP=OLD,DCB=BUFNO=40,                          03220000
//             DSN=1111111.CCUMSMTC                                     03230000
//SORTOUT      DD       DISP=OLD,DCB=BUFNO=40,                          03240000
//             DSN=1111111.CCUMSMTC                                     03250000
//SYSIN        DD       *                                               03260000
  SORT FIELDS=(5,1,CH,A,1,4,CH,A)                                      03270000
//SYSOUT       DD       SYSOUT=*                                        03280000
//*-----------------------------------------------------------------*/  03290000
//ERRORLOC    EXEC     PGM=IKJEFT01,DYNAMNBR=30,                        03300000
//             COND=((4,LT),(4,NE,HSUMCOMP),(0,NE,COMPARE))             03310000
//SYSTSPRT     DD       SYSOUT=*                                        03320000
//SYSTSIN      DD       *                                               03330000
 DSN SYSTEM(2222222)                                                   03340000
 RUN PROGRAM(EKYC600X) PLAN(3333333)                                   03350000
//SYSPRINT     DD       SYSOUT=*                                        03360000
//SYSUDUMP     DD       SYSOUT=*                                        03370000
//EKYRESLB     DD       DISP=SHR,                                       03380000
//             DSN=1111111.EKYRESLB                                     03390000
//CCUPRINT     DD       SYSOUT=*                                        03400000
//CCUCDS       DD       DISP=SHR,                                       03410000
//             DSN=1111111.CCUCDS                                       03420000
//CCUMSMTC     DD       DISP=OLD,                                       03430000
//             DSN=1111111.CCUMSMTC                                     03440000
//CCUDERR      DD       DISP=(,CATLG,CATLG),                            03450000
//             SPACE=(TRK,(5,5),RLSE),                                  03460000
//             DSN=1111111.CCUDERR                                      03470000
//CCUKEY1      DD       DISP=OLD,                                       03480000
//             DSN=1111111.CCUKEY1                                      03490000
//CCUKEY2      DD       DISP=OLD,                                       03500000
//             DSN=1111111.CCUKEY2                                      03510000
//EKYTRACE     DD       SYSOUT=*                                        03520000
//CCUDBOUT     DD       DISP=OLD,                                       03530000
//             DSN=1111111.CCUDBOUT                                     03540000
//*-----------------------------------------------------------------*/  03550000
//*            SORT DB2 REPAIR STATEMENT FILE                        */  03560000
//*-----------------------------------------------------------------*/  03570000
//SORTDB2     EXEC     SORT,                                            03580000
//             COND=((4,LT),(4,NE,HSUMCOMP),(0,NE,COMPARE))             03590000
//SORTIN       DD       DISP=OLD,DCB=BUFNO=40,                          03600000
//             DSN=1111111.CCUDERR                                      03610000
//SORTOUT      DD       DISP=OLD,DCB=BUFNO=40,                          03620000
//             DSN=1111111.CCUDERR                                      03630000
//SYSIN        DD       *                                               03640000
  SORT FIELDS=(74,7,CH,A)                                              03650000
//SYSOUT       DD       SYSOUT=*                                        03660000
```

Figure 305. Sample JCL for EKYUCCHD (Part 6 of 6)

DD Statements

The DD statements //STEPLIB and //EKYRESLB are common to many IMS DPROP components and are described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3. The remaining DD statements included in the job steps of the CCU are:

//ASYSIN DD Statement

Used for the PSB generation in the initialization phase, it replaces the //SYSIN DD statement of the assembler.

//ASYSLIB DD Statement
Used for the PSB generation in the initialization phase, it replaces the //SYSLIB DD statement of the assembler.

//ASYSLIN DD Statement
Used for the PSB generation in the initialization phase, it replaces the //SYSLIN DD statement of the assembler.

//ASYSPRT DD Statement
Used for the PSB generation in the initialization phase, it replaces the //SYSPRINT DD statement of the assembler.

//CCUCDS DD Statement
The internal control block data set. It is created in the initialization phase and is used as input to all the other phases.

//CCUDBD DD Statement
The IMS DBDLIB that contains the IMS DBD load module for the database that should be checked. It is used in the initialization phase.

//CCUDBIN DD Statement
Optional. Used in an asynchronous IMS DPROP system to replace the IMS database with the name of an HD unload file in the format generated by the DFSURGU0 HD unload utility. The file is input to the IMS read phase of the hashing technique. See *IMS/ESA Utilities Reference: System* and *IMS/ESA Utilities Reference: Database Manager* for more information on DFSURGU0 and "Using the CCU in an Asynchronous IMS DPROP System" on page 419.

//CCUDBOUT DD Statement
Optional. Used in an asynchronous IMS DPROP system if the IMS data comes from the HD unload file. //CCUDBOUT is output from the IMS read phase of the hashing technique, and input to the error location phase. See Table 37 on page 418 for information on calculating space allocation.

//CCUDERR DD Statement
Names the file that contains the DB2 repair statements. It is created by the error location phase of the hashing technique, or the read and compare phase of the direct technique if the READONLY keyword is specified.

//CCUHASH1 DD Statement
Contains the hashed IMS segments and is used in the compare phase. It is created by the IMS read phase of the hashing technique, if the HASHONLY keyword was not specified. See Table 37 on page 418 for information on calculating space allocation.

//CCUHASH2 DD Statement
Contains the hashed DB2 rows and is used in the compare phase. It is created by the DB2 read phase of the hashing technique, if the HASHONLY keyword was not specified. See Table 37 on page 418 for information on calculating space allocation.

//CCUHSUM1 DD Statement
Names the file that contains the total hash sums for the IMS segments. It is created by the IMS read phase if you are using the hashing technique, and used in the hash sum compare phase. See Table 37 on page 418 for information on calculating space allocation.

//CCUHSUM2 DD Statement
Names the file that contains the total hash sums for the DB2 rows. It is created

by the DB2 read phase if you are using the hashing technique, and used in the hash sum compare phase. See Table 37 on page 418 for information on calculating space allocation.

//CCUIERR DD Statement
A data set that names the files that contains the IMS repair statements. It is created by the error location phase or the read and compare phase of the direct technique if the READONLY keyword is specified.

//CCUIN DD Statement
Contains your input control statement. It is input to the initialization phase.

//CCUKEY1 DD Statement
Names the VSAM data set created by the IMS read phase if you are using the hashing technique, and if you did not specify the HASHONLY keyword. It contains the mapped, concatenated key fields of the IMS segments. This file is used in the error location phase. See Table 37 on page 418 for information on calculating space allocation.

//CCUKEY2 DD Statement
Names the VSAM data set created by the DB2 read phase if you are using the hashing technique, and if you did not specify the HASHONLY keyword. It contains the primary key columns of the DB2 rows. This file is used in the error location phase. See Table 37 on page 418 for information on calculating space allocation.

//CCUMSMTC DD Statement
Names the data set that contains the records that indicate mismatches between IMS and DB2. In the direct technique, this file is only created if you did not specify the READONLY keyword during the initialization phase. In the hashing technique, this file is created in the compare phase. It is used as input to the error location phase.

//CCUPRINT DD Statement
Contains the CCU messages. It is used in all CCU phases.

//CCUSORTI DD Statement
Used in the compare phase, it concatenates the two data sets, //CCUHASH1 and //CCUHASH2, which are created in the IMS and DB2 read phases if you are using the hashing technique.

//CCUSORTS DD Statement
Names the data set created by the hash sum compare phase if you are using the hashing technique. It contains the sort statement used in the compare phase. See Table 37 on page 418 for information on calculating space allocation.

//database DD Statement
Names the IMS database that the CCU should check when the CCU runs in a BMP region, or, optionally, in an asynchronous IMS DPROP system. If you use dynamic database allocation, you can omit this statement.

//DFSRESLB DD Statement
Names the authorized library containing the IMS load modules. It is used in the IMS read phase of the hashing technique, in the read and compare phase of the direct technique and in the error location phase. If your installation uses dynamic allocation for the IMS RESLIB, you can omit this statement.

//IMS DD Statement
Used in IMS job steps to allocate the IMS DBDLIB and PSBLIB. If your

installation uses dynamic allocation for the IMS DBDLIB and PSBLIB, or if the CCU runs in an IMS BMP region, you can omit this statement.

//IMSACB DD Statement
The output data set for the ACBGEN job step.

//index DD Statement
Names the IMS database primary index part if the database organization form is an indexed one (for example, HIDAM, HISAM). If you use dynamic database allocation, if the CCU runs in a BMP region, or, optionally, in an asynchronous IMS DPROP system, you can omit this statement.

//SORTCNTL DD Statement
An input data set used in the compare step. It contains the OMIT or EXCLUDE statement, which is generated by the hash sum compare phase and stored in the //CCUSORTS data set.

//SORTOUT DD Statement
Used for the sort step in the compare phase. It contains the mismatch indications.

//SORTWKxx DD Statement
The temporary work data sets used in the sort step of the compare phase.

//SYSOUT DD Statement
Used for the sort step in the compare phase.

//SYSIN DD Statement
Used by the DB2 TSO TMP environment, the ACBGEN, the CCUKEY1 and CCUKEY2 cluster allocation, and the compare step,

//SYSLMOD DD Statement
Used for the PSB generation in the initialization phase. After the initialization phase ends, it contains the generated PSB load module.

//SYSPRINT DD Statement
Used to print messages.

//SYSUT1 and //SUSUT2 DD Statements
Used for the PSB generation in the initialization phase.

//SYSTSIN DD Statement
Used by the DB2 TSO TMP environment.

//SYSTSPRT DD Statement
Used by the DB2 TSO TMP environment.

The Initialization Phase

During the initialization phase, the CCU:

- Accepts and analyzes input control statements and, if necessary, prints error messages and stops the run.
- Reads the IMS DPROP Directory.
- Generates the CCU internal control blocks in the //CCUCDS data set.
- Compares user input control statements with internal control blocks and, if necessary, prints error messages and stops the run.
- Creates a PSB load module for the IMS database to be checked. See “Generated PSB Source Code” on page 470.
- Creates dynamic SQL statements for the DB2 tables to be checked. See “Generated SQL Statements ” on page 472.

The PSBNAME= keyword of the CHECK control statement generates a PSB during CCU initialization.By using this PSB, you avoid errors that occur if missing SENSEGs prevent the PSB from retrieving necessary segments. Missing SENSEGs in the PSB result in a large CCU error file. The CCU interprets the missing SENSEGs as missing segment occurrences in the IMS database. Also, by using this PSB, you avoid overhead caused by retrieving segments that do not need to be retrieved.

For asynchronous propagation, however, generating a PSB is optional if you use an IMS HD unload file as IMS data input for the CCU. If you do not want to generate a PSB, you must modify the JCL for the initialization phase. See Figure 305 on page 439 for a JCL sample that does not generate a PSB. By creating its own PSB during initialization, with all the SENSEGs that are required during the subsequent read and error location phases, the CCU prevents possible error files and reduces overhead while checking segments.

For example, the read and error-location phases require a valid IMS PCB within a PSB. Depending on the SENSEGs in the PSB, the *corresponding* segment types are read. If you wanted to check a segment type that you did not previously define in the PSB, the CCU would not retrieve this segment. This could result in a large error file, even though no errors exist.

Additionally, when all possible SENSEGs are in the PSB but you only want to check one segment, the CCU would retrieve all segments. To prevent this type of overhead, the CCU creates its own PSB with all required SENSEGs. See “Using the CCU in an Asynchronous IMS DPROP System” on page 419 for information how to submit the CCU without PSB generation.

You can run the CCU in an IMS Batch or in a BMP region. If you run the CCU in a BMP region, you must:

- Define an IMS APPLCTN macro with DOPT parameter for the CCU PSB. for more information on DOPT, see IMS/ESA Installation Volume 2: System Definition and Tailoring.
- Submit an ACBGEN for the PSB after the CCU initialization phase.

  The created ACB must reside in a library other than the primary IMSVS.ACBLIB and must be concatenated to it. See Figure 307 on page 449.
- Ensure that the DB2 plan name of the read phase and error location phase match the CCU load module names, EKYC100X, EKYC300X, and EKYC600X. Alternatively, you can use a resource translation table (RTT).

Input and Output

Figure 306 on page 449 illustrates the sources of input to the CCU and the output created by the CCU during initialization.

Input and Output During Initialization Phase

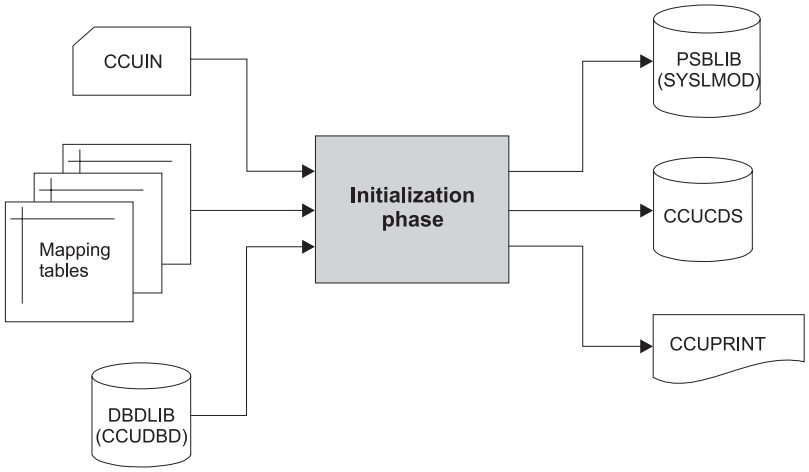

Figure 306. Input and Output during Initialization Phase

JCL for the Initialization Phase

Figure 302 on page 426 and Figure 304 on page 432 show the JCL for the initialization phase.

If the CCU runs in an IMS BMP region, an ACBGEN is required. Figure 307 shows the JCL to perform an ACBGEN.

```
//ACBGEN       EXEC     PGM=DFSRRC00,PARM='UPB'
//SYSPRINT     DD       SYSOUT=A
//DFSRESLB     DD       DISP=SHR,DSN=IMSVS.RESLIB /* Change        */
//IMS          DD       DISP=SHR,DSN=IMSVS.PSBLIB/* as required    */
//             DD       DISP=SHR,DSN=IMSVS.DBDLIB/*                 */
//IMSACB       DD       DISP=SHR,DSN=IMSVS.ACBLIB/*                 */
//SYSIN        DD       *
       BUILD PSB=ccupsb
```

Figure 307. Sample ACBGEN for Running the CCU in an IMS BMP Region

For information on the DD statements in all of the CCU job steps, see “DD Statements ” on page 444.

Output Messages and Statistics

The prefix for messages from the initialization phase start is EKYC0. The CCU writes messages to the CCUPRINT file. For more information on messages, see*IMS DPROP Messages and Codes*.

Return Codes and Error Conditions

The initialization phase of the CCU provides the following return codes:

Code	Meaning
0	Initialization ended successfully. Input control statements have been accepted, the PSB load module and CCUCDS data set have been created.
4	The same conditions as for return code 0, but there was at least one warning message.
8	There was at least one error. The initialization phase must be started again after solving the error.

For more information on CCU messages, see *IMS DPROP Messages and Codes.*.

The Read and Compare Phase

During the read and compare phase, the CCU:

- Initializes the CCU control blocks.
- Reads the IMS database.
- Reads the related DB2 tables.
- Compares the propagated fields of IMS segments with the corresponding columns of the related DB2 tables.
- Creates output files whether it detected data inconsistencies or not.

Rules for Field and Column Comparison

The CCU processes the IMS fields and DB2 columns according to the mapping rules described in the appropriate Administrators Guide for your propagation mode. There are some special mapping considerations which are explained in the description of message EKYC816I on page 494.

Input and Output Using Read and Compare (Direct)

When you use the direct technique, the IMS segments are compared with the DB2 rows as they are read. Input and output to the IMS and DB2 read phase is shown in Figure 308 on page 451.

Input and Output for IMS and DB2 Read Phase (Direct)

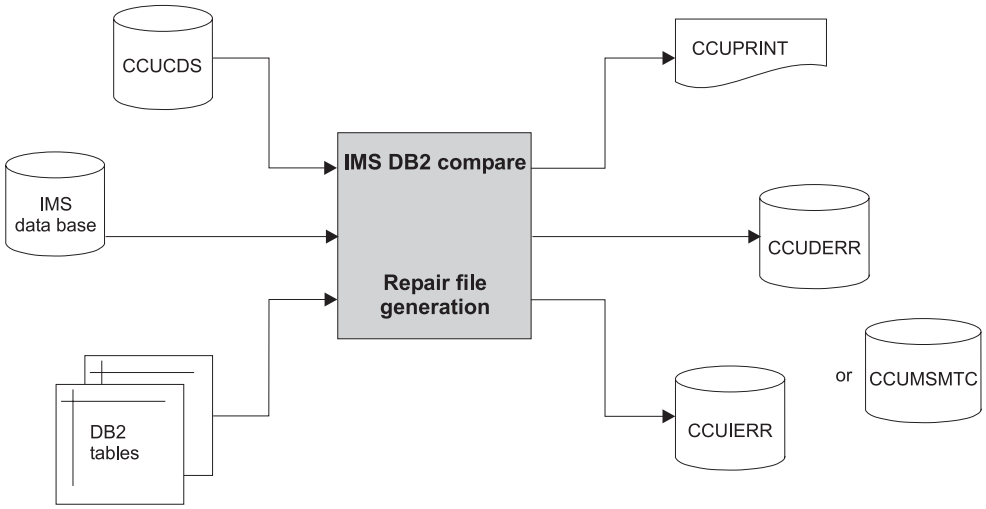

Figure 308. Input and Output for IMS and DB2 Read Phase (Direct)

If no mismatches are found, the CCU execution stops at the end of the read and compare phase. What happens when mismatches *are* found depends on whether you specify the READONLY keyword in the CHECK input statement:

- If you do *not* specify READONLY, the CCU assumes that:
  - The mismatches are the consequence of updates running concurrently to the CCU execution
  - The mismatches are temporary

  Therefore, the CCU writes the mismatches to the mismatch indication file (CCUMSMTC), and passes the file to the error location phase to verify the mismatches once again.
- If you *do* specify READONLY, the CCU assumes that the two data copies are in read-only status and that the mismatches are not the consequence of concurrent updates. The read and compare phase, therefore, creates repair statements for potential repair actions. The IMS and/or DB2 repair statements are generated and written to the //CCUIERR and //CCUDERR files. The CCU execution stops at the end of the read and compare phase.

For more information on the READONLY keyword, see “The Check control statement ” on page 460.

Input and Output Using IMS or DB2 Read Step

When the initialization phase is complete, you can start the IMS and DB2 read phase. You can submit these two phases as two job steps of the same job, or in parallel, as independent jobs.

Input during the IMS and DB2 read phases includes the //CCUCDS data set, the IMS database or HD unload file, and the DB2 tables.

Output messages are sent to the //CCUPRINT file. Other output includes the //CCUKEYn, //CCUHASHn, and //CCUHSUMn files.

- The //CCUKEY1 and //CCUKEY2 files contain the mapped key fields of the IMS database and the key columns of the DB2 tables, respectively. They are used in the error location phase if you did not specify HASHONLY on the CHECK control statement.
- The //CCUHASH1 and //CCUHASH2 files contain the hashed IMS fields and hashed DB2 columns of the IMS database and DB2 tables, respectively. They are used in the compare phase if you did not specify HASHONLY on the CHECK control statement.
- The //CCUHSUM1 and //CCUHSUM2 files contain the sums of the hashed values of the IMS database and DB2 tables, respectively, and are used during the hash sum compare phase.

If the IMS segments come from the HD unload file or the DEDB Unload file (//CCUDBIN), the output file //CCUDBOUT is created.

IMS Read Step

In the IMS read phase, the CCU reads the required IMS segments from either the IMS database or an HD unload file as it is created by the IMS HD Reorganization Unload utility, DFSURG0.

If the CCU retrieves the IMS segments from an IMS database, the CCU reads all the segments specified in the PSB and marks the ones that must be checked. For segments retrieved from the IMS database, the CCU either does not write to an output file, or writes to one or more output files, according to your control statements. Figure 309 demonstrates input and output for the IMS read phase.

Input and Output for the IMS Read Phase using the IMS Database

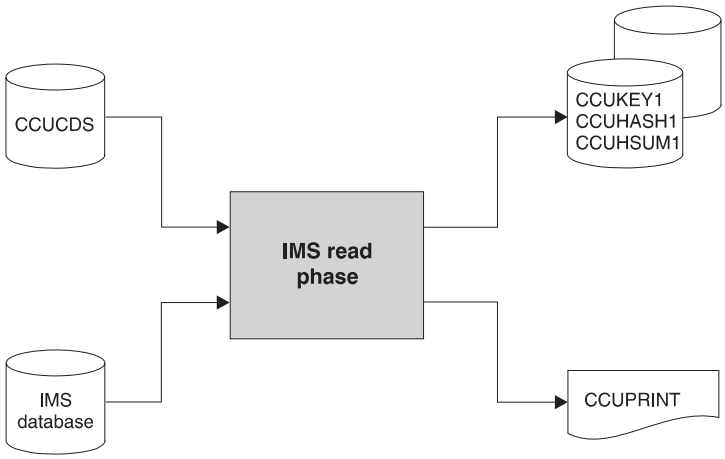

Figure 309. Input and Output for the IMS Read Phase using the IMS Database

If the CCU retrieves the IMS segments from an HD unload file, the CCU creates an additional output file, CCUDBOUT. This file contains the mapped IMS segments in an internal format. It is input to the error location phase. Figure 310 on page 453 demonstrates the use of the DDUDBOUT output file.

Input and Output for the IMS Read Phase using the HD Unload File

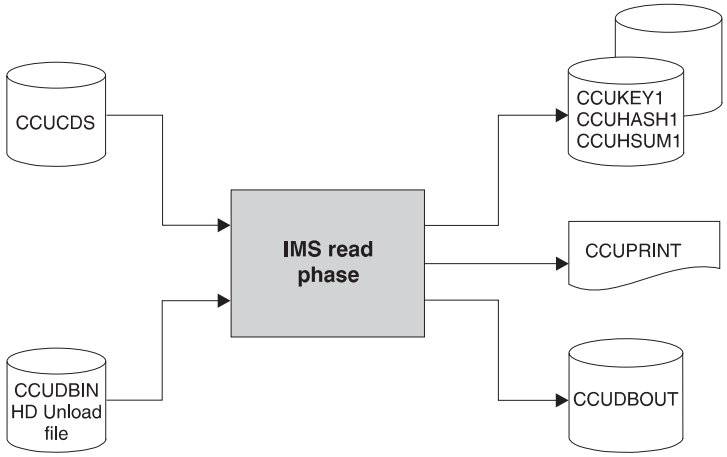

Figure 310. Input and Output for the IMS Read Phase using the HD Unload file

For more information on asynchronous propagation, see “Using the CCU in an Asynchronous IMS DPROP System” on page 419. For more information on DFSURGU0, see *IMS/ESA Utilities Reference: System* and *IMS/ESA Utilities Reference: Database Manager*.

DB2 Read Step

In the DB2 read phase, the CCU hashes the retrieved rows the same way it does for the IMS segments. Depending on your output statements the CCU writes to zero, one or more output files. Figure 311 illustrates the input and output for the DB2 read phase.

Input and Output for DB2 Read Phase

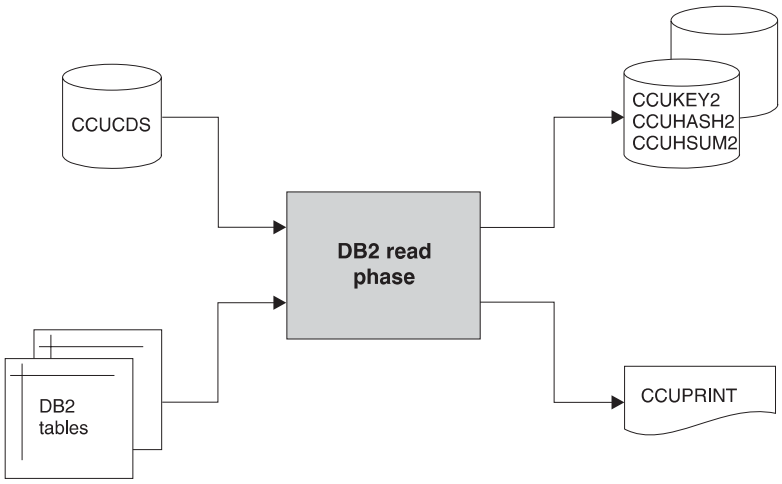

Figure 311. Input and Output for DB2 Read Phase

Hash Sum Compare Phase

During the hash sum compare phase, the CCU compares the hash sum files that were created during the IMS and DB2 read phases, //CCUHSUM1 and //CCUHSUM2. Figure 312 illustrates the input and output to the hash sum compare phase.

Input and Output for Hash Sum Compare Phase

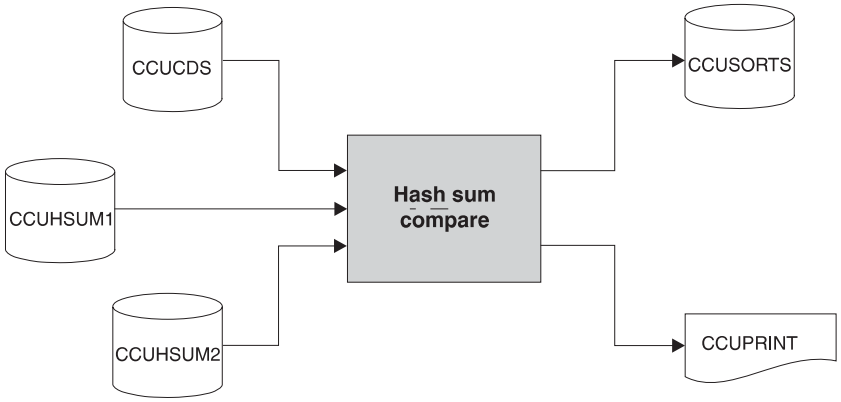

Figure 312. Input and Output for Hash Sum Compare Phase

If the hash sums are equal, the CCU assumes that the IMS and DB2 copies are consistent and the compare phase is not required. If the hash sums do not match, the IMS and DB2 copies are inconsistent. Then compare and error location phases are required to locate, verify, and identify the errors.

Compare Phase

During the compare phase, the CCU compares the IMS segments and DB2 records using the //CCUHASH1, //CCUHASH2, //CCUKEY1 and //CCUKEY2 files that were generated during the read phases. It eliminates all matched pairs, leaving only the mismatches. Figure 313 on page 455 shows the input and output to the compare phase.

Input and Output for the Compare Phase

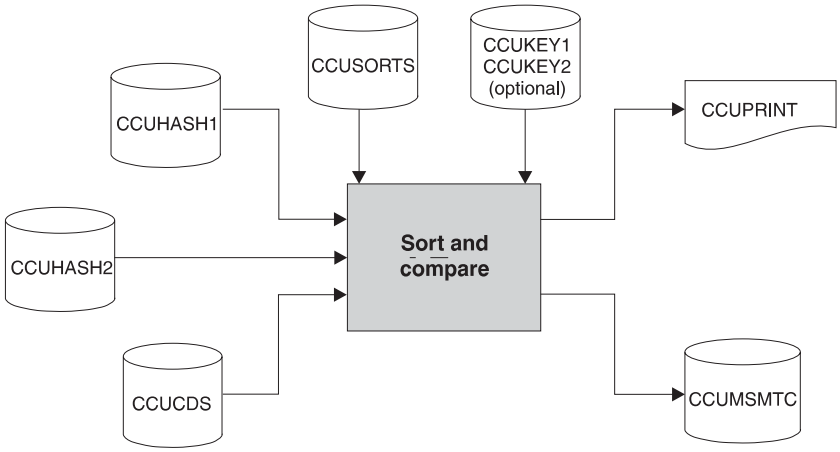

Figure 313. Input and Output for the Compare Phase

The CCU writes the mismatches remaining from the compare phase to //CCUMSMTC, which is required for the error location phase.

JCL Requirements

See Figure 302 on page 426 for the JCL to use the IMS/DB2 direct technique read phase if you did not specify the READONLY keyword on the CHECK control statement during the initialization phase. Figure 303 on page 430 shows the JCL for the IMS/DB2 direct technique read phase if you used the READONLY keyword during the initialization phase.

Figure 304 on page 432 shows the JCL for the IMS and DB2 read phase in an IMS batch region using the hashing technique. Figure 304 on page 432 also contains sample JCL for the hash sum compare phase and for the compare phase of the hashing technique.

Note that the compare phase should only be executed if the hash sum compare phase ends with a return code 4, indicating that the two data copies do not match.

Output Messages and Statistics

Messages from the read and compare phases are as follows:

Step	Message begins with
IMS Read (hashing)	EKYC1
DB2 Read (hashing)	EKYC2
IMS and DB2 Read (direct)	EKYC3
Hash Sum Compare (hashing)	EKYC4
Compare (hashing)	EKYC5
additional messages for any of above steps	EKYC8 EKYC9

Messages from the read and compare phases are written to the CCUPRINT file. For more information on messages, see *IMS DPROP Messages and Codes*, or Chapter 19, “Interpreting CCU Reports,” on page 467.

Return Codes and Error Conditions

This topic describes the return codes for IMS and DB2 for the read and compare phases.

Read And Compare Phase (Direct)

The IMS and DB2 read and compare phase of the direct technique provides the following return codes:

Code	Meaning
0	The read phase ended successfully. No CCU internal error was detected; all mismatch messages are written to the //CCUPRINT file. If you did not specify KEYONLY on your control statement, the repair statements are written to the //CCUDERR and //CCUIERR files (if you specified READONLY) or the //CCUMSMTC file (if you did not specify READONLY). This code is returned whether or not data mismatches occur.
4	Same as for code 0, but at least one warning message was printed.
8	The read phase terminated with errors. See the generated messages in the CCUPRINT file and the explanations in *IMS DPROP Messages and Codes*.

IMS Read Phase (Hashing)

The IMS read phase of the hashing technique provides the following return codes:

Code	Meaning
0	The read phase for the IMS database ended successfully. No CCU internal error was detected, and no unexpected IMS status code was returned.
4	Same conditions as code 0, but at least one warning message was printed.
8	The read phase for the IMS database terminated with errors. See the generated messages in the CCUPRINT file and the explanations in *IMS DPROP Messages and Codes*.

DB2 Read Phase (Hashing)

The DB2 read phase of the hashing technique provides the following return codes:

Code	Meaning
0	The read phase for the DB2 tables ended successfully. No CCU internal error was detected, and no unexpected DB2 SQL code was returned.
4	Same conditions as code 0, but at least one warning message was issued.
8	The read phase for the DB2 tables terminated with errors. See the generated messages in the CCUPRINT file and the explanations in *IMS DPROP Messages and Codes*.

Hash Sum Compare Phase (Hashing)

The hash sum compare phase of the hashing technique provides the following return codes:

Code	Meaning
0	The hash sum compare phase ended successfully. No CCU internal error

was detected. The hash sums in //CCUHSUM1 match those in //CCUHSUM2. Therefore, the CCU assumes that the IMS and DB2 data are consistent.

4 The hash sum compare phase ended successfully. No CCU internal error was detected. However, the hash sums in //CCUHSUM1 do *not* match those in //CCUHSUM2. The IMS and DB2 copy are probably inconsistent. Execute the compare phase and the error location phase to verify and identify the mismatches.

8 The hash sum compare phase terminated with errors. See the generated messages in the CCUPRINT file and the explanations in*IMS DPROP Messages and Codes*.

Compare Phase (Hashing)

The compare phase of the hashing technique provides the following return codes:

Code	Meaning
0	The compare phase ended successfully. No CCU internal error was detected. All mismatches are written to the //CCUMSMTC file.
4	The compare phase ended successfully. No CCU internal error was detected. However, the data in //CCUHASH1 matches that in //CCUHASH2; the CCU therefore assumes that the IMS and DB2 data is consistent.
8	The compare phase terminated with errors. See the generated messages in the CCUPRINT file and the explanations in*IMS DPROP Messages and Codes*
16	A sort error occurred.

The Error Location Phase

During the error location phase the CCU uses mismatch indication records created during previous CCU phases to verify and identify data inconsistencies in the IMS and DB2 data copies. Tasks the CCU performs during the error location phase are:

- Read the IMS segment occurrences and DB2 rows identified in the //CCUMSMTC file.
- Determine whether the IMS and DB2 copies of the data identified in //CCUMSMTC match.
- Write error messages when it detects mismatches.
- Create repair statements for the segment or row in error, if you did not specify the KEYONLY keyword.

You can execute this phase only if you did not specify the READONLY or HASHONLY keywords during the initialization phase.

The detected mismatches are in the //CCUMSMTC file, in the form of hashed values of the rows and segments.

Input and Output

Figure 314 on page 458 shows the CCU input and output during the error location phase.

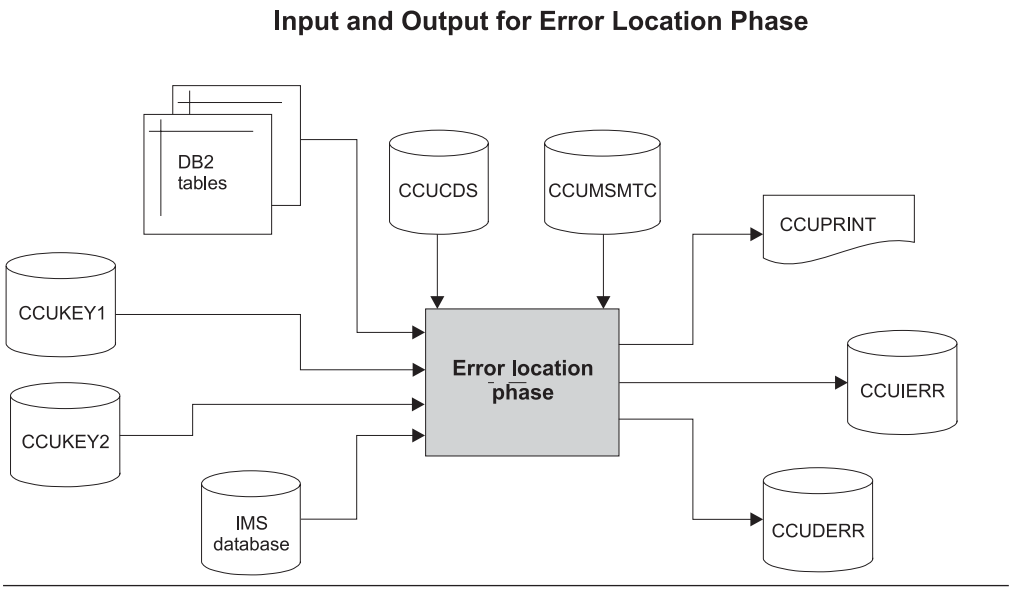

Figure 314. Input and Output for Error Location Phase

The IMS and DB2 read phases generate //CCUKEY1 and //CCUKEY2 respectively.

If you are using asynchronous propagation and submit the IMS read phase with an IMS HD unload file or a DEDB unload file as input, the error location phase uses the //CCUDBOUT file created during the IMS read phase instead of the IMS DB as its input. //CCUDBOUT contains the mapped IMS segments in an internal format. For more information on using the HD Unload file, see “Using the CCU in an Asynchronous IMS DPROP System” on page 419.

Output from the error location phase consists of:
- //CCUPRINT, which contains any messages
- //CCUIERR, which contains IMS repair statements
- //CCUDERR, which contains DB2 repair statements

For more information on how the repair data set is generated, see to “Repair Data Set Generation ” on page 459.

JCL Requirements

See Figure 302 on page 426 and Figure 304 on page 432 for the JCL for the error location phase.

If you use an HD unload file in the IMS read phase, see the JCL for the error location phase in Figure 305 on page 439.

For information on the DD statements in all of the CCU job steps, see “DD Statements ” on page 444.

If you submit the hashing technique, the error location phase runs only if the hash sum compare phase resulted in a return code 4 and the compare phase resulted in a return code 0.

Output Messages and Statistics

Messages from the error location phase are sent to the //CCUPRINT file.

For more information on messages, see*IMS DPROP Messages and Codes* and Chapter 19, “Interpreting CCU Reports,” on page 467.

Return Codes and Error Conditions

The error location phase of the CCU run provides the following return codes:

Code	Meaning
0	The error location phase ended successfully.
4	The error location phase ended successfully; at least one warning message was printed.
8	The error location phase terminated with errors. See the generated messages in the CCUPRINT file and the explanations in*IMS DPROP Messages and Codes*.

Repair Data Set Generation

If the CCU locates any errors during either the error location phase of the hashing technique, or during the read and compare phase of the direct technique, it generates repair data sets, //CCUDERR and //CCUIERR. The CCU first writes the errors to the //CCUPRINT file, and then generates //CCUDERR and //CCUIERR. You can use these files to help you repair mismatches.

DB2 Repair Data Set //CCUDERR

The CCU only generates SQL repair statements if the propagation requests PRTYPE are limited, full, or extended function (PRTYPE=L, F or E). and if the mapping direction is hierarchical-to-relational or two-way (MAPDIR=HR or TW). See the appropriate Administrators Guide for your propagation mode for more information.

The //CCUDERR repair data set contains SQL statements. A typical DB2 repair data set might contain:
- A qualified UPDATE statement if a non-key data mismatch occurred
- An INSERT statement if a DB2 row was missing
- A qualified DELETE statement if an IMS segment was missing
- The contents of the propagated column for the corresponding IMS segment for generated UPDATE and INSERT statements

For more information on using the repair statements, see “Repair Data Set for the DB2 Tables” on page 506.

IMS Repair Data Set //CCUIERR

DL/I repair statements are generated only if the propagation request PRTYPE is extended-function (PRTYPE=E). and if the mapping direction is relational-to-hierarchical or two-way (MAPDIR=RH or TW).

No DL/I repair statements are created for an asynchronous IMS DPROP system.

The //CCUIERR repair data set contains DL/I calls. A typical DL/I repair data set might contain:
- A qualified REPL call if a non-key data mismatch occurred
- An ISRT call if an IMS segment was missing
- A qualified DLET call if a DB2 row was missing

Some DL/I calls include a preceding Get Hold Unique (GHU) call to position the effective repair call under its required parent segment.

For more information on using the repair statements, see “Repair Data Set for the IMS Database ” on page 508.

The Check control statement

Use the CHECK control statement to identify the IMS database, IMS segments, DB2 tables, or propagation requests to be checked.

This topic describes some rules to consider when using the CCU and the CHECK control statement.

General Rules

The CCU does not support checking multiple DB2 tables for each IMS segment within one run, except:
- Segments propagated by propagation requests defined with a WHERE clause
- Segments involved in a mapping case 3 propagation request

If you do check multiple DB2 tables for each IMS segment type, conflicting IMS and DB2 repair statements could result. Analyze the generated repair files carefully before applying them to the IMS and/or DB2 data.

Rules for Mapping Case 2

The following rule applies to mapping case 2 extension segments that have propagated dependent segments. In one execution, the CCU can check either:
- The extension segments of the mapping case 2 propagation requests
- The propagated dependent segments of the extension segments.

If you do not specify which one to check, the CCU checks the propagated dependent segments. This is the default.

Rules for Mapping Case 3

The READONLY keyword is not supported for mapping case 3 propagation requests,

For each IMS segment type within one CCU submission you can check either:
- The mapping case 3 propagation requests propagating the internal segments of this IMS segment type
- The propagation request that propagates the IMS segment as the entity (mapping case 1 or 2)

If the same data portion of an IMS segment is propagated by multiple propagation requests (field overlapping), specify an ASSIGN keyword that includes only one of the request IDs. If you do not specify an ASSIGN keyword the CCU might create invalid repair statements for the physical IMS segment if data mismatches occur. The exceptions to this rule are:
- Data portions located in the IMS key portion
- Data located in the containing IMS segment and propagated by the mapping case 3 propagation requests as path data

To create a list of propagation request IDs that propagate an IMS segment type, query the IMS DPROP directory, as shown in Figure 315.

```
SELECT  B.DBNAME, A.PRSET, B.SEGNAME, B.ROLE, B.MAPCASE, B.PRID
   FROM DPRPR A, DPRSEG B
  WHERE A.PRID    = B.PRID
    AND B.DBNAME  = 'IMS database name'
    AND B.SEGNAME = 'IMS segment type'
    AND B.ROLE IN ('C','E')
  ORDER BY A.PRSET
```

Figure 315. IMS DPROP Directory Query

Syntax of the CHECK control statement

Figure 316 on page 462 shows the syntax of the CHECK control statement.

```
►►─CHECK──┬─DBD=dbdname──┬─ Segment block ─┬──┬─ Assign block ─┬──┬─ PRSET block ─┬──►
          │              │                 │  │                │  │               │
          │              └─────────────────┘  └────────────────┘  └───────────────┘
          │
          ├─TAB=──(──┬──────────────┬──tabname──┬──)──────────────────────────────►
          │          │ ,            │           │
          │          │              │           │
          │          └──tabqualifier.──────────┘
          │
          │
          └─PR──(──┬──prid──┬──)──────────────────────────────────────────────────►
                   │ ,      │
                   │        │
                   └────────┘

►──┬──────────────────────────────────────────┬──┬──────────────────────────────►
   │                                          │  │
   │                                          │  │
   │  ┌──────────────────────────────────┐    │  └─,QUALIFIER=tabqual2──┘
   │  │                                  │    │
   └──┴──,USE=(──tabname,viewname──)──────┘    │

►──┬──────────────────────────────────────────┬──┬──────────────────────────────►
   │                                          │  │
   ├─,DIRECT──┬──────────┬──┬──────────┬──┬──┬──┤  └─,KEYONLY──┘  └─,MAXERROR=n──┘
   │          │ ,FORCE   │  │ ,READONLY│  │  │  │
   │          └──────────┘  └──────────┘  │  │  │
   │                                      │  │  │
   └─,HASHONLY────────────────────────────┘  │  │

►──┬──────────────────────────────────────────────────────────────────────────┬──;──►◄
   │                                                                          │
   └─,PSBNAME=psbname──┬──────────────────────┬──┬──────────────────────┬──┬──┘
                       │ ,PROCOPT=procopt     │  │ ,KEYLEN=pcbkeylen    │  │
                       └──────────────────────┘  └──────────────────────┘  │
```

Segment block:

```
├──,──┬─SEG=─────┬──(──┬──────────┬──segname──┬──)──────────────────────────────┤
      │          │     │ ,        │           │
      └─EXCLUDE=─┘     │          │           │
                       └──────────┘
```

Assign block:

```
├──,ASSIGN=──(──┬──────────────┬──segname=prid──┬──)──────────────────────────────┤
                │ ,            │                │
                │              │                │
                └──────────────┘
```

PRSET block:

```
├──,PRSET=prset──────────────────────────────────────────────────────────────────┤
```

Figure 316. The CHECK Statement

You can provide only one CHECK control statement for each CCU execution. One CCU execution cannot check propagation requests belonging to different PRSETs.

One CCU execution cannot check multiple propagation requests propagating the same segment type, except mapping case 3 propagation requests and propagation requests defined with a WHERE clause.

DBD= *dbdname*
Identifies the database to check. Use any propagated physical DBD name. Unless you specify a SEG= or EXCLUDE= keyword, all segments of the named

database propagated by propagation requests that are supported by the CCU are compared with the data content of the related DB2 tables.

SEG=(*segment1,segment2,...* **)**
Identifies the IMS segments to compare with the related DB2 tables. You can list one or more segment names on this statement.

Use this keyword if you want to check only specific segment types instead of all propagated segments from the physical DBD.

For mapping case 2 propagation requests you can identify the entity segment type alone, or the entity segment type and one or more extension segment types. You cannot specify extension segments without their entity segment.

For mapping case 3 segments, you must specify the physical IMS segment name; that is, the containing segment name, not the internal segment name.

EXCLUDE=(*segment1,segment2,...* **)**
Identifies segments of an IMS database that you do not want to check. You can list one or more segment names on this statement.

For mapping case 2 propagation requests, you can exclude one or more extension segment types or the entity segment type. If you exclude the entity segment, all its extension segments are excluded also.

For mapping case 3 segments, you must enter the IMS segment name, not the internal segment name. The external segments are excluded.

ASSIGN=(*segment1=prid7,segment2=prid4,...* **)**
Specifies which propagation requests the CCU should check for a specific IMS segment type when there is more than one propagation request for an IMS segment type. The CCU does not check multiple propagation requests for an IMS segment, except mapping case 3 propagation requests and propagation requests defined with a WHERE clause.

For mapping case 2 propagation requests, use the ASSIGN= keyword for entity segments only. Extension segments, if selected, are automatically assigned the same propagation request as their entity segment.

For mapping case 3 propagation requests, you must enter the IMS segment name, not the internal segment name.

PRSET= *prset*
Identifies the PRSET to process if the same IMS database is propagated by propagation requests belonging to multiple PRSETs. The CCU only processes propagation requests of one PRSET for each execution.

TAB=(*table1,tabqual.table2,...* **)**
Names the DB2 tables to compare with the related IMS segments. The table name must be the same as the DB2 table name specified when the propagation request was coded. It can be a qualified table name, a two part table name consisting of a DB2 qualifier and a name, or an unqualified table name consisting of one part. Separate the qualifier and the name with a period.

You can list one or more table names in the TAB statement. However, you must propagate each table name in a list of names from or to the same IMS database as propagation requests of the same PRSET. You cannot propagate two table names in the list from or to the same IMS segment type, except propagation requests defined with a WHERE clause, or mapping case 3 propagation requests.

PR=(*prid1,prid2,...* **)**
Identifies the propagation requests to be checked. *prid* can be any existing propagation request name.

You can list one or more *prids* in the propagation request keyword. However, each propagation request in the list must propagate the same IMS database and belong to the same PRSET. Two propagation requests in the list cannot propagate the same IMS segment type except propagation requests defined with a WHERE clause, and mapping case 3 propagation requests.

USE=(*table,view* **)**
Identifies the table and view during the DB2 read phase if you want the CCU to select the DB2 rows from a specific view instead of a DB2 table.

The USE= keyword can be useful when you have segment exit routines that suppress propagation.

The table name must be the DB2 qualifier and DB2 table name specified when the propagation request was coded. Separate the prefix and the name with a period. The view name must be a DB2 view on the associated table; it must include all propagated columns from the table. The view name cannot refer to more than one table, however, the CCU does not check this. You cannot use GROUP BY or HAVING clauses within the view definition.

You can use qualified or unqualified names.

The column names specified in the CREATE VIEW statement must match the column names specified in the CREATE TABLE statement. For more information on these statements, see DB2 SQL Reference.

QUALIFIER= *tabqual2*
The qualifier used for DB2 table names if the DB2 table name found in the IMS DPROP directory is blank. The value of *tabqual2* qualifies only table names that are not in a USE keyword. It can be an alphanumeric value, with a maximum length of 8 bytes.

DIRECT
In a synchronous IMS DPROP system, it specifies the CCU is to use the direct technique instead of the hashing technique. If you do not specify DIRECT, the CCU defaults to the hashing technique.

In the direct technique, the CCU creates dynamic SQL statements with an ORDER BY clause for every DB2 table it checks. In the hashing technique, the CCU creates dynamic SQL statements, but without the ORDER BY clause. The ORDER BY clause retrieves the DB2 rows in the same order as the IMS segment occurrences. The ORDER BY clause lists the columns of the DB2 primary key. The sequence of the DB2 columns within the ORDER BY clause is determined by the location of the corresponding IMS fields within the IMS segment hierarchy and within the segments. The criteria of the ORDER BY column sequence is:
- The hierarchy of the IMS segments
- The start position of the IMS field within every segment

For each column, the ASC/DESC attribute is the same in the ORDER BY clause as in the primary key index.

FORCE
By default, the CCU rejects use of the direct technique when it is most likely not applicable. Specifying the FORCE keyword to force the CCU to accept the DIRECT keyword and use the direct technique.

If you use the FORCE keyword when the direct technique is not valid, the CCU creates a list of data inconsistencies where none likely exists.

Some examples of valid usage of the FORCE keyword are:

- You are comparing segments from an HDAM database *and* your DBD uses an HDAM sequential randomizer.

  For a descending ordering sequence, if you know that the retrieve sequence of the IMS segments corresponds to the retrieve sequence of the DB2 row, specify the FORCE keyword to force the CCU to use the direct technique. However, the FORCE keyword is not recommended if you are checking an HDAM database without a sequential randomizer.
- The CCU is not accepting the combination of the DIRECT and READONLY keywords. For example, a DB2 primary key column of a propagated table has a descending ordering sequence, or it has a data type of DECIMAL, INTEGER, or SMALLINT.

  For DB2 columns with the numeric datatypes, if you know that the data values of these fields and columns are positive numbers *only*, specify the FORCE keyword to force the CCU to use the direct technique.

READONLY

In most cases, however, you should analyze the generated repair statements before applying them to the IMS or DB2 data. See 464 for more information and some consequences of using the FORCE keyword. If used with the DIRECT keyword, READONLY tells the CCU that the IMS database and the DB2 tables to be checked are in read-only status. This enables the DB2 and IMS repair data sets to be created in the CCU read phase of the direct technique. You will do need to submit the error location phase to create the repair data sets. The CCU does not verify whether or not the IMS and DB2 data really *are* in a read-only status.

You cannot specify READONLY if you are checking data propagated by a mapping case 3 propagation request.

HASHONLY

With the hashing technique, the data content of the segments and the related DB2 rows is analyzed, but the files //CCUKEYn and //CCUHASHn, required to submit the CCUs compare and error location phase, are not created. HASHONLY can shorten the run time of the CCU. If you replace the IMS database with the HD unload file, the HASHONLY keyword does not save elapsed time in the IMS read phase.

After the hash sum compare phase, the CCU stops processing. The CCU writes messages to the //CCUPRINT output data set identifying the propagation request IDs whose data is consistent and those whose data is not consistent. If there are inconsistencies, you must rerun the CCU *without* the HASHONLY keyword.

KEYONLY

Suppresses the analysis of the non-key fields within the IMS segments and non-key columns of the DB2 rows. It checks the presence of the IMS segments and the related propagated DB2 rows only. The CCU notifies you if mismatches occur.

The CCU does not create //CCUKEYn and //CCUHASHn, which are required to submit the CCUs repair file generation. If there are inconsistencies and you want the repair files to be generated, you must rerun the CCU *without* the KEYONLY keyword.

You cannot use KEYONLY for propagation requests defined with path data, or if a non-key IMS field is mapped to the DB2 primary key.

If you replace the IMS database with the HD unload file, the KEYONLY keyword will not save elapsed time in the IMS read phase, because the HD unload file must be read totally.

You can combine the KEYONLY and HASHONLY keywords.

You can combine the KEYONLY and the DIRECT keywords. If you do and mismatches occur, only the erroneous keys are reported on the //CCUPRINT output data set. The repair files are not generated.

MAXERROR= *n*
The number of data inconsistencies that the CCU is to process. See the description of message EKYC630W in Figure 364 on page 493 for more details on this subject.

n is any number between 1 and 9999, where 9999 is *all* errors. The default for MAXERROR is 100.

PSBNAME= *psbname*
The name under which the PSB load module must be stored in the allocated PSB library. The name can be up to 8 bytes, and must start with an alphabetic character. A PSB name is required in a synchronous and optional in an asynchronous IMS DPROP system.

PROCOPT= *procopt*
The IMS processing option for the PSB generated by the CCU. The default is *G*. The CCU uses the PSB during the read and error location phases.

KEYLEN= *pcbkeylen*
Overwrites the IMS database PCB key length if you do not use the length generated by the CCU.

Chapter 19. Interpreting CCU Reports

In many of the CCU phases, you receive messages prompting you to analyze the processing and the output of the CCU. This topic helps you understand how the CCU processes data and describes the information and warning messages that the CCU generates.

If the CCU does not detect an error or no error message is generated, you can use the reports that the CCU creates to determine how to handle data inconsistencies. Each of the seven CCU phases generates reports. The CCU also generates repair data sets when it detects data mismatches.

The following topics provide additional information:

Initialization Phase Reports

The initialization phase produces the following reports:
- Start messages
- Specified user input control statement
- Information and warning messages
- Generated PSB source code
- Generated SQL statements
- Control data set timestamp
- End message

Initialization Phase Start Messages

```
 DPROP120 PGM=EKYC000X JOB/STEP=JOB1 INIT DATE=93.001 TIME=12.00.00  DDN=CCUPRINT PAGE= 1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC000I INITIALIZATION PHASE STARTED
```

Figure 317. Initialization Phase Start Messages

The messages in Figure 317 indicate the IMS DPROP system name used when the CCU was submitted, DPROP001. An initial started message is also listed.

EKYX116I and its accompanying text indicates that the CCU program started reading the IMS DPROP directory master table. The mapping table qualifier of the IMS DPROP system was specified during DPROPGEN.

Specified User Input control statement

Figure 318 shows the messages that precede the user input control statement. The CCU prints the user input control statements that you specified in the //CCUIN input data set.

```
EKYC010I CHECK STATEMENT ENTERED BY THE USER FOLLOWS THIS MESSAGE:

          CHECK DBD=PRODDB,
                   SEG=(S0000,S1000,S1300,S3000,S5100),
                   ASSIGN=(S5100=PR5100A,S5100=PR5100B),
                MAXERROR=100,
                PSBNAME=EKYCCU00, PROCOPT=G
              ;
```

Figure 318. Specified User Input control statement Message

Initialization Phase Information and Warning Messages

Information and warning messages during the CCU initialization phase can be caused by:

- Not specifying a PRSET= keyword with your CHECK DBD= keyword on the //MVGPARM data set, or specifying a CHECK PR= or CHECK TAB= keyword. You then receive message EKYC047I indicating which IMS database and PRSET the CCU is going to process, as illustrated in Figure 319:

```
EKYC047I CCU USES DBNAME PRODDB PRSET NAME PRSET1, RECEIVED FROM THE DPROP DIRECTORY
```

Figure 319. Message EKYC047I

- Having at least two mapping case 3 PRIDs referring to the same IMS segment type. You then receive message EKYC059I, shown in Figure 320.

```
EKYC059I TWO PRIDS, PR5100A AND PR5100B, REFER TO SEGMENT S5100.
         THE CCU WILL PROCESS BOTH PRIDS, BUT IN CASE OF DATA INCONSISTENCIES,
         THE REPAIR STATEMENTS FOR THESE PRIDS MIGHT BE CONFLICTING
```

Figure 320. Message EKYC059I

For example, in the mapping case 3 example in Figure 320:

1. There is a non-key data mismatch for a specific S5100 segment in the database, and the related row in the target table.
2. The CCU generates an IMS REPL call for the mismatch.
3. The CCU processes PR5100B, and for that IMS segment occurrence, the DB2 row is missing from the target table.
4. The CCU generates IMS DLET for this data mismatch.
5. Now there are two conflicting repair statements: one to update a non-key data mismatch within PRID PR5100A, and one to delete the segment for PRID PR5100B.

In this situation, you usually receive message EKYC070I or EKYC072I, shown in Figure 321 on page 469, describing the PRIDs, segments and DB2 table names that refer to a specific PRSET within the selected IMS database. The list that follows the message includes a CHECK column indicating whether or not a specific PRID is checked through the current CCU job stream, and some additional COMMENTS, if applicable.

```
EKYC070I PRSET PRSET1 OF DATABASE PRODDB CONTAINS THE FOLLOWING PRIDS, SEGMENTS AND PROPAGATED TABLES:
EKYC072I DATABASE PRODDB CONTAINS THE FOLLOWING PRIDS, SEGMENTS AND PROPAGATED TABLES:

                 PR-  MAPPING-
         PRID    TYPE CASE DIR SEGMENT  ROLE PARENT   DB2 TABLE NAME              CHECK COMMENTS
         -------- ---- ---- --- -------- ---- -------- --------------------------- ----- ---------------------------
         PR0000   E    1    TW  S0000    E             DB2PROD.T0000               YES   IN SEG= LIST
         PR1000   E    2    TW  S1000    E    S0000    DB2PROD.T1000               YES   IN SEG= LIST
         PR1000   E    2    TW  S1300    X    S1000    DB2PROD.T1000               YES   IN SEG= LIST
         PR1100   E    1    TW  S1100    E    S1000    DB2PROD.T1100               NO
         PR2000   E    1    TW  S2000    E    S0000    DB2PROD.T2000               NO
         PR3000   E    1    TW  S3000    E    S0000    DB2PROD.T3000               YES   IN SEG= LIST
         PR4000   E    1    TW  S4000    E    S0000    DB2PROD.T4000               NO
         PR5000   E    1    TW  S5000    E    S0000    DB2PROD.T5000               NO
         PR5100   E    2    TW  S5100    E    S5000    DB2PROD.T5100               NO    IN ASSIGN= LIST, OTHER PRID
         PR5100   E    2    TW  S5101    X    S5100    DB2PROD.T5100               NO
         PR5100A  E    3    TW  S5100    C    S5000    DB2PROD.T5100A              YES   IN SEG= AND ASSIGN= LIST
         PR5100A  E    3    TW  S5100A   E    S5100    DB2PROD.T5100A              YES   INTERNAL SEGMENT
         PR5100B  E    3    TW  S5100    C    S5000    DB2PROD.T5100B              YES   IN SEG= AND ASSIGN= LIST
         PR5100B  E    3    TW  S5100B   E    S5100    DB2PROD.T5100B              YES   INTERNAL SEGMENT
         PR5100C  E    3    TW  S5100    C    S5000    DB2PROD.T5100C              NO    IN ASSIGN= LIST, OTHER PRID
         PR5100C  E    3    TW  S5100C   E    S5100    DB2PROD.T5100C              NO    INTERNAL SEGMENT
```

Figure 321. Message EKYC070I

- Generating or revalidating a specific PRID. When you do so the CCU generates message EKYC046I, shown in Figure 322. The message is followed by a list of PRIDs and messages that were generated by the MVG utility when it generated or revalidated a PRID.

  You can use these messages to help detect the reason for data inconsistencies the CCU detects during the read or error location phase. See *IMS DPROP Messages and Codes* for descriptions of the MVG utility messages. If you want to verify whether a printed message is valid, you can submit an MVG REVALIDATE job for the PRIDs listed in the message.

```
EKYC046I THE FOLLOWING MESSAGES WERE GENERATED WHEN MVG VALIDATED THE PRS:

         PRID     MESSAGE  TEXT
         -------- -------- ---------------------------------------------------------------
         PR3000   EKYM170W THE FULLY CONCATENATED KEY OF THE ENTITY SEGMENT IS UNIQUE;
                           MAPPING TO THE DB2 PRIMARY KEY SHOULD BE DONE EXCLUSIVELY BY
                           IT; DBD=PRODDB, SEG=S3000
```

Figure 322. Message EKYC046I

- Using both positive and negative numbers when mapping an IMS numeric field to a DB2 numeric data type, when using the direct technique. A DB2 column defined as packed decimal can have numeric values with either a sign X'C' (positive number), or X'D' (negative number). Packed fields in an IMS database can have positive numbers with the signs X'A', X'C' or X'F', and negative numbers with the signs X'B' and X'D'. The message you receive is EKYC069W, shown in Figure 323 on page 470.

  Therefore, even if the CCU tries to receive the DB2 rows in an ascending sequence (using the ORDER BY clause in the SQL statements), the DB2 rows cannot always be retrieved in the same sequence that the IMS segments are retrieved.

  If your database contains IMS segment types having packed decimal fields in the segment key, determine whether or not these key fields can contain negative numbers. If they can, do not use the CCU direct technique; instead, use the hashing technique. If you are sure that the numbers will only be positive, disregard message EKYC069W.

```
EKYC069W CCU DATA MISMATCH INDICATIONS CAN BE CAUSED BY COMBINING THE DIRECT TECHNIQUE
         WITH KEY MAPPING A NUMERIC IMS FIELD TO A DB2 PRIMARY KEY COLUMN HAVING A NUMERIC
         DATATYPE IF THE VALUES OF THESE FIELDS/COLUMNS CAN CONTAIN BOTH POSITIVE AND
         NEGATIVE NUMBERS

         PRID     SEGMENT  FIELD    DATATYPE DB2 COLUMN NAME    COLTYPE  DB2 TABLE NAME
         -------- -------- -------- -------- ------------------ -------- -----------------
         PR4000   S4000    K4000    PACKED   K4000              DECIMAL  DB2PROD."T4000"
```

Figure 323. Message EKYC069W

Generated PSB Source Code

During the first part of the initialization phase, the CCU retrieved information from the IMS DPROP mapping tables and the IMS DBDLIB. Now it generates, assembles and links PSB source code and writes it to the //CCUPRINT data set, shown in Figure 324.

```
EKYC085I THE FOLLOWING PSB SOURCE CODE IS ASSEMBLED AND LINKED INTO IMSVS.PSBLIB:

         CCUPCB   PCB   TYPE=DB,SB=COND,DBDNAME=PRODDB,PROCOPT=G,              C
                        PCBNAME=CCUPCB,KEYLEN=00017
                  SENSEG NAME=S0000,PARENT=0
                  SENSEG NAME=S1000,PARENT=S0000
                  SENSEG NAME=S1300,PARENT=S1000
                  SENSEG NAME=S3000,PARENT=S0000
                  SENSEG NAME=S5000,PARENT=S0000,PROCOPT=K
                  SENSEG NAME=S5100,PARENT=S5000
                  PSBGEN LANG=ASSEM,PSBNAME=EKYCCU00,                          C
                        IOEROPN=(451,WTOR),CMPAT=YES
                  END
```

Figure 324. Generated PSB Source Code Messages

The PSB consists of the following four statements:

- PCB statement

  The PCB statement, shown in Figure 324, uses the same value for both the PCB label and name. The PCB label allows allocation of additional buffers for sequential buffering. The PCB name is required for DFS™ application interface block (DFSAIB) processing. Keywords for the PCB statement are:

  SB=COND
  Allows IMS to issue sequential buffering for this PCB.

  DBDNAME=PRODDB
  - Names the physical IMS DBD
  - Identifies the IMS DBD load module allocated on the //CCUDBD DD statement.

  PROCOPT=G
  The processing option for accessing the IMS database.

  KEYLEN=00017
  The key length of the longest path of the IMS database to be checked.

- SENSEG statement

  The sensitive segments (SENSEGs) are listed in hierarchical sequence. The list contains the segment names of all segments that are either:

  - To be checked

– The parent in the physical path of the segment that must be checked. In this case, the SENSEG statement in the PSB source code is:

```
SENSEG NAME=segmname,PARENT=prntname,PROCOPT=K
```

- PSBGEN statement

  The PSBGEN statement contains the name of the PSB load module (EKYCCU00), and identifies how it is stored in the PSBLIB you allocated through the //SYSLMOD DD statement.

  CMPAT=YES allows the PSB to be used in an IMS batch (BPP) as well as batch message processing (BMP) region.

- END statement

For more information on PSB source code, refer to*IMS/ESA Utilities Reference: System*

PSB Concerns for the CCU

Table 40 describes how you must code the PSB to be used by the CCU, and what happens if you fail to do so.

Table 40. Rules for CCU Processing Based on PSB Source Code

Rule Number	Rule	Action if Rule is Violated
1	PSB must be composed of one DB PCB, one or more SENSEGs, the PSBGEN, and the END statement.	The CCU issues message EKYC831W.
2	DBDNAME in PCB statement must be a physical DBD.	• If the DBD is logical, only segment types defined in the logical DBD are retrieved. • If the DBD name describes an indexed DBD (for example), the read and error location phase terminates after receiving an unexpected DL/I status code.
3	PCB statement must contain PCBNAME=CCUPCB.	The CCU terminates processing with message EKYC860E.
4	DB PCB must be composed of one or more SENSEG statements.	The CCU does not retrieve the segments of the missing segment type. The corresponding rows on the DB2 side are read, creating an immense mismatch report and unnecessary repair statements.
5	All segments to be checked must be defined in a SENSEG statement.	Same as Rule 4.
6	All parent segments in the physical path, including the segment to be checked, must be defined in a SENSEG statement.	Same as Rule 4.
7	A segment type that will not be checked, but is in the physical path of a segment to be checked, should have a PROCOPT=K keyword (exception: if the segment type has source for PATH data or is mapped through a WHERE clause).	The segment retrieval process will have a higher CPU/elapsed time.

Table 40. Rules for CCU Processing Based on PSB Source Code (continued)

Rule Number	Rule	Action if Rule is Violated
8	If you specified KEYONLY on the CHECK control statement, do not define mapping case 2 extension segments in the DB PCB.	Same as Rule 7.
9	Do not specify PROCOPT=K for a segment type that the CCU should check.	The CCU does not retrieve the segment's data, and creates an immense error list for this segment type.
10	SENSEG statements must not have SENFLD statements.	The CCU retrieves only the fields specified in the SENFLD statements and passes to the RUP an invalid segment length. The RUP issues an error message, and the CCU terminates.
11	CMPAT=YES must be defined in the PSBGEN statement.	The CCU terminates processing after it receives an unexpected DL/I status code when trying to process a DL/I INQY call.
12	The PSBNAME keyword must be the same PSB name defined in the CHECK control statement.	The CCU issues message EKYC833I.
13	The language must be ASSEM.	The CCU terminates with message EKYC830E.
14	A PROCSEQ= keyword must not be used for the DB PCB that accesses the DL/I segments if: • The sequence of the keys from the retrieved segments through that PROCSEQ keyword no longer match the DB2 key sequence. • The processing sequence is such that not all segment types in a CCU-relevant segment path can be retrieved.	CCU results will be unpredictable. The generated DB2 repair data set must not be used.

If you are using the direct technique with the READONLY keyword and processing option G, the READONLY keyword is applicable when all data actually *is* read only. If you choose READONLY and specify a DB PCB processing option of "GO," for example, and parallel update jobs modify the data content of the DL/I database, the CCU might report data inconsistencies where none exist. If that happens, the repair statements *falsely* update rows of a DB2 table and lead to CCU-generated data mismatches.

Generated SQL Statements

After the CCU generates the PSB load module, it generates the SQL statements for the DB2 read phase and the error location phase. The CCU writes the SQL statements that retrieve the data in the read and compare phases to the //CCUPRINT data set.

The EKYC095I messages in the report identify:

- The propagation request ID

- The related DB2 table name with its qualifier, if the IMS DPROP mapping table contains a qualifier for the named DB2 table

In Figure 325, the DB2 table name is imbedded in double quotation marks (″). If your system administrator specified a single quotation mark (’) as the DB2 delimiter, then use a single quote is in the table name also.

```
EKYC095I SQL STATEMENT FOR PRID PR0000, TABLE <DB2PROD."T0000">:
         SELECT "K0000", "D0000" FROM DB2PROD."T0000"

EKYC095I SQL STATEMENT FOR PRID PR5100A, TABLE <DB2PROD."T5100A">:
         SELECT "K0000", "K5000", "C5000", "C5001", "C5002", "K5100",
            "K5100VG" FROM DB2PROD."T5100A"
```

Figure 325. Message EKYC095I: Generated SQL Statements

The generated SQL statement is printed after the message. It contains:

- The column names, in sequence, as they were found in column COLNO of the IMS DPROP mapping table DPRFLD.
- The DB2 table qualifier and table name, separated by a period.

The keywords you use with the CHECK control statement, affect the generated SQL statements. For example:

- If you use the DIRECT keyword, the SQL statements are generated with an ORDER BY list, as shown in Figure 326. The list contains all the DB2 columns that are defined as DB2 primary keys. “K0000” is defined as the DB2 primary key column of table DB2PROD.″T0000″.

```
EKYC095I SQL STATEMENT FOR PRID PR0000, TABLE <DB2PROD."T0000">:
         SELECT "K0000", "D0000" FROM DB2PROD."T0000" ORDER BY "K0000"
```

Figure 326. Message EKYC095I with the DIRECT Keyword

- If you use the KEYONLY keyword on the CHECK control statement, the SQL statements contain only the DB2 columns that are defined as DB2 primary keys. Figure 327 shows that only “K0000” is in the list, because it is the only column defined as the DB2 primary key of DB2PROD.″T0000″.

```
EKYC095I SQL STATEMENT FOR PRID PR0000, TABLE <DB2PROD."T0000">:
         SELECT "K0000" FROM DB2PROD."T0000"
```

Figure 327. EKYC095I with KEYONLY Keyword

- If you use the USE keyword on the CHECK control statement, the SQL statement contains the DB2 view name DB2PROD.″T0000_VIEW″ as illustrated in Figure 328, instead of the table name DB2PROD.″T0000″.

```
EKYC095I SQL STATEMENT FOR PRID PR0000, TABLE <DB2PROD."T0000_VIEW">:
         SELECT "K0000", "D0000" FROM DB2PROD."T0000_VIEW"
```

Figure 328. EKYC095I with USE Keyword

- If you submit the CCU using the DIRECT keyword, but not the READONLY keyword on the CHECK control statement, you might receive message EKYC096I, which indicates data mismatches. However, the error location phase will probably nullify most of the data mismatch indications.

- If you submit the CCU with the DIRECT *and* the READONLY keywords, you might receive message EKYC097W, shown in Figure 329, which indicates that the CCU has generated repair files that you should analyze. The repair files are probably not applicable, and *must not be applied* to the IMS or DB2 data.

  In this situation, you might prefer to submit the CCU hashing technique. You can also determine whether or not you want to recreate the propagation request with the MVGU using a different value for the KEYORDER keyword. For more information, refer to the description of the KEYORDER keyword in Chapter 5, “//MVGPARM Data Set Propagation Parameters,” on page 133.

```
EKYC096I COMBINING THE CCU DIRECT TECHNIQUE WITH A DESCENDING ORDERING SEQUENCE OF DB2
         PRIMARY KEY COLUMNS COULD LEAD TO A HUGE DATA MISMATCH INDICATION LIST IN THE
         READ AND COMPARE PHASE

         PRID     SEGMENT  FIELD    DB2 TABLE                  COLUMN             ORDERING
         -------- -------- -------- -------------------------- ------------------ --------
         PR4000   S4000    K4000    DB2PROD."T4000"            K4000                  DESC

EKYC097W COMBINING THE CCU DIRECT TECHNIQUE AND THE DESCENDING ORDERING SEQUENCE OF DB2
         PRIMARY KEY COLUMNS COULD LEAD TO GENERATED REPAIR FILES THAT SHOULD BE CAREFULLY
         ANALYZED BEFORE APPLYING THEM TO THE IMS AND/OR DB2 DATA

         PRID     SEGMENT  FIELD    DB2 TABLE                   COLUMN             ORDERING
         -------- -------- -------- --------------------------- ------------------ --------
         PR4000   S4000    K4000    DB2PROD."T4000"             K4000                  DESC
```

Figure 329. Message EKYC096I and EKYC097W

Segments that you access in an IMS database are normally returned to you an key-ascending sequence. However, the related DB2 tables can return the rows to you in other than key-ascending sequence. If you use the direct technique, the CCU assumes the DB2 rows are in the same key-sequence as the IMS segments. If they are not, you receive a data mismatch indication list for the PRIDs involved. Therefore, you must use the ORDER BY clause to specify the key sequence for DB2 to guarantee it matches the sequence used by IMS.

Control Data Set Timestamp

Figure 330 shows message EKYC002I, which indicates the timestamp and the selected processing technique. The CCU prints this message at the end of the initialization phase report.

```
EKYC002I CONTROL DATA SET SUCCESSFULLY CREATED.
         TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING HASHING TECHNIQUE.
         DPROP SYSTEM NAME=DB2PROD  TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS
```

Figure 330. Control Data Set Timestamp

You can submit the CCU multiple times, both in parallel and against the same IMS database and DB2 tables. Therefore, a unique key must be created for every CCU job stream you submit. The timestamp serves as the unique key, representing the start time of the CCU initialization phase.

The timestamp is set into the control data set, which acts as input to the other CCU phases after initialization. It is set into all other output files generated by the CCU. This enables the CCU to check whether the data sets belong together during subsequent processing phases.

Initialization Phase End Messages

Figure 331 shows the final CCU initialization phase messages, EKYC003I and EKYC004I:

```
EKYC003I INITIALIZATION PHASE ENDED NORMALLY

EKYC004I INITIALIZATION PHASE ENDED WITH ERRORS
```

Figure 331. Initialization Phase End Messages

- The CCU writes EKYC003I when the CCU initialization phase ends successfully. The control data set (//CCUCDS) is generated, and the PSB load module is stored in the library you allocated with the //SYSLMOD DD statement.

  If you receive a warning message, you should determine what caused the message, correct it, and resubmit the step.
- The CCU writes EKYC004I if any error occurred. The //CCUCDS is not created, and the remaining CCU steps cannot be submitted. Scan the initialization phase reports for error messages and determine the cause of error.

IMS Read Phase Reports for the Hashing Technique

If you selected the hashing technique to check the data, then the next step is the IMS read phase. During the read phase the CCU retrieves the segments from the database and creates files that *represent* the content of the data being propagated and checked by this run.

The IMS read phase of the hashing technique produces the following reports:
- Start messages
- Information and warning messages
- Segment statistics
- End messages

IMS Read Phase Start Messages

Message EKYX116I provides the name of the IMS DPROP system used when the CCU IMS read phase was submitted, and indicates the CCU has started reading the IMS DPROP directory master table. In Figure 332 the name of the system is DPROP001. You also receive message EKYC100I, which indicates the read phase has started.

```
 DPROP120 PGM=EKYC100X JOB/STEP=JOB1 G IMSREAD DATE=93.001 TIME=12.00.00 DDN=CCUPRINT PAGE=1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC100I DL/I READ PHASE STARTED
```

Figure 332. IMS Read Phase Start Messages

The mapping table qualifier of the IMS DPROP system was specified during DPROPGEN.

IMS Read Phase Information and Warning Messages

Most of the information and warning messages issued in the IMS read phase are common to the messages issued by both the read and compare phase of the direct technique and the error location phase. Information and warning messages of the IMS read phase might include:

```
EKYC815I EKYC821I EKYC825I EKYC828I EKYC831W
EKYC833I EKYC862W EKYC865W EKYC866W EKYC906W
```

Refer to “Common Messages for All CCU Phases ” on page 490 for descriptions of the messages.

IMS Read Phase Segment Statistics

You receive message EKYC868I after the IMS read phase has processed all segments from the IMS database. If the program found an error, you receive message EKYC869I, followed by the segment statistic report, shown in Figure 333.

```
EKYC868I STATISTICS FOR RETRIEVED DL/I SEGMENTS OF DATABASE PRODDB:
EKYC869I STATISTICS FOR RETRIEVED DL/I SEGMENTS OF DATABASE PRODDB AT TIME OF ERROR:

                                                SEGMENT TYPE IS PROPAGATED BY       NUMBER OF
         TYPE    RETRIEVED   CHECK   PRID     MAPPING CASE/OPTIONS   WITH ROLE  RESULTING ROWS
         -----------------   -----   ---------------------------------------- --------------
         S0000           4   YES     PR0000   MC1                    ENTITY                 4
                                     PR5100A  MC3 / PATHDATA         PARENT                 -
                                     PR5100B  MC3 / PATHDATA         PARENT                 -
         S1000           4   YES     PR1000   MC2                    ENTITY                 4
         S1300           2   YES     PR1000   MC2                    EXTENSION              4
         S2000           5   NO      N/A
         S3000           9   YES     PR3000   MC1                    ENTITY                 9
         S4000           0   YES     PR4000   MC1 / WHERE            ENTITY                 0
         S5000           4           PR5100A  MC3 / PATHDATA         PARENT                 -
                                     PR5100B  MC3 / PATHDATA         PARENT                 -
         S5100           6   YES     PR5100A  MC3 / PATHDATA         CONTAINING             -
         S5100A              YES     PR5100A  MC3                    ENTITY                10
                             YES     PR5100B  MC3 / PATHDATA         CONTAINING             -
         S5100B              YES     PR5100B  MC3                    ENTITY                 5
```

Figure 333. Segment Statistics

The following list describes the four parts of the segment statistic report:

TYPE and RETRIEVED

The number of segments retrieved for each IMS segment type.

If no segments of a specific segment type are retrieved, and you specified your own PSB instead of using the CCU-generated PSB, check whether the segment type with ’0’ occurrences is defined in your PSB.

The names S5100A and S5100B in Figure 333 are not real IMS segment types; they are internal segments of mapping case 3 PRIDs.

CHECK

The CHECK column indicates whether the segment is selected to be checked:

YES The segment type is selected.

NO The segment type is not selected, but it is in the physical path to another segment type that has to be checked. If the segment type is not in the physical path to another segment that is selected, then retrieving these segments consumes elapsed time and is needless overhead.

blank The segment type was used to map a dependent segment type with PATH data or a WHERE clause.

PRID NAME

The PRID name and some of the propagation request characteristics.

NUMBER OF ROWS RESULTING
Indicates how many of the retrieved segments the RUP converted to the format of a DB2 row.

IMS Read Phase End Messages

The IMS read phase ends with one of the messages shown in Figure 334 :

```
EKYC102I DL/I READ PHASE ENDED AT END OF DATA
EKYC102I DL/I READ PHASE ENDED AFTER 100 ERRORS

EKYC103I DL/I READ PHASE ENDED WITH ERRORS
```

Figure 334. IMS Read Phase End Messages

- EKYC102I indicates the IMS read phase ended successfully. The data sets required for subsequent CCU job steps have been generated successfully.

  With this message, you can receive a message indicating "WARNING MESSAGES HAVE BEEN ISSUED." If you do, determine what caused the warning message, correct it and resubmit the step. The CCU continues processing subsequent steps.
- EKYC103I indicates if any error occurred. Scan the read phase reports for error messages and correct any errors. The CCU does not create the data sets required by subsequent CCU job steps, so you cannot submit the subsequent steps.

DB2 Read Phase Reports for the Hashing Technique

If you are using the hashing technique to check the data, then you submit the DB2 read phase next. This phase retrieves the rows from all DB2 tables to be checked and creates files that *represent* the content of the data being propagated and being checked by this run.

The DB2 read phase generates the following reports for the hashing technique:
- Start messages
- Information and warning messages
- Table statistics
- End messages

DB2 Read Phase Start Messages

Messages EKYX116I and EKYC200I provide the IMS DPROP system name that the DB2 read phase is using and an initial started message. In Figure 335 the IMS DPROP system name is DPROP001.

```
DPROP120 PGM=EKYC200X JOB/STEP=JOB1  DB2READ  DATE=93.001 TIME=12.00.00 DDN=CCUPRINT PAGE=1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC200I DB2 READ PHASE STARTED
```

Figure 335. DB2 Read Phase Start Messages

Message EKYX116I indicates that the CCU program started reading the IMS DPROP directory master table. If you need the mapping table qualifier of the specified IMS DPROP system, your system administrator can identify who specified the qualifier during DPROPGEN.

DB2 Read Phase Information and Warning Messages

Most of the information and warning messages issued in the DB2 read phase are common to the messages issued by the read and compare phase of the direct technique, and to the messages issued by the error location phase.

Information and warning messages of the DB2 read phase include:

```
EKYC815I EKYC818I EKYC820I EKYC821I EKYC824W
EKYC825I EKYC828I EKYC855I EKYC862W EKYC865W
```

Refer to “Common Messages for All CCU Phases ” on page 490 for their descriptions.

DB2 Read Phase Table Statistics

After the DB2 read phase processes all DB2 rows from the tables to be checked, the CCU generates the table statistic report, shown in Figure 336. The report contains message EKYC875I or EKYC876I, and indicates the number of rows retrieved for every processed DB2 table.

```
EKYC875I STATISTICS FOR RETRIEVED ROWS FROM DB2 TABLES:
EKYC876I STATISTICS FOR RETRIEVED ROWS FROM DB2 TABLES AT TIME OF ERROR:

          ROWS RETRIEVED DB2 TABLE NAME                PRID     MAPPING CASE / OPTIONS
         --------------- ---------------------------- -------- ----------------------
                       4 DB2PROD."T0000"              PR0000   MC1
                       4 DB2PROD."T1000"              PR1000   MC2
                       7 DB2PROD."T3000"              PR3000   MC1
                       0 DB2PROD."T4000_VIEW"         PR3000   MC1
                      10 DB2PROD."T5100A "            PR5100A  MC3 / PATHDATA
                       6 DB2PROD."T5100B "            PR5100B  MC3 / PATHDATA
```

Figure 336. Table Statistics

If the program did not retrieve at least one row from a table, use one of the following methods to determine why no rows were retrieved:

- If you coded a USE keyword on the CHECK control statement, check the definition of the view.
- If the table name is unqualified, DB2 implicit qualification can have invalidly qualified the table. If so, determine whether the program accessed the table you wanted to check. Refer to Figure 368 on page 497 for more information on DB2 implicit qualification.

As shown in Figure 333 on page 476, message EKYC869I lists the segment statistics for the IMS database and the mapping case 2 extension segments. In Figure 336, table DB2PROD.″T1000″ includes the data of extension segment S1300 in the table statistic but does not show it explicitly.

Figure 336 shows that DB2PROD.″T3000″, which is propagated-to by segment S3000, displays seven retrieved rows. However, Figure 333 on page 476 shows that nine segments of segment S3000 were retrieved in the IMS read phase. If no segment exit routine is active, or no WHERE clause is defined for the propagation request, you can assume that 2 rows are missing on the DB2 table, and the data is probably inconsistent. Non-key data mismatches could occur between segment S0000 and table DB2PROD.″T0000″, but you cannot determine that by comparing the IMS and DB2 statistics.

DB2 Read Phase End Messages

The DB2 read phase ends with one of the messages shown in Figure 337:

```
EKYC202I DB2 READ PHASE ENDED AT END OF DATA

EKYC203I DB2 READ PHASE ENDED WITH ERRORS
```

Figure 337. Message EKYC297I

- EKYC202I indicates the IMS read phase ended successfully. The data sets required for subsequent CCU job steps have been generated successfully.

  With this message, you might also receive a message that "WARNING MESSAGES HAVE BEEN ISSUED." If you do, determine what caused the warning message, correct it, and resubmit the step. However, the CCU continues processing subsequent steps.
- EKYC203I indicates an error occurred. Scan the read phase reports for error messages and correct the errors. The data sets required by subsequent CCU job steps were not created, so the subsequent steps cannot be submitted.

Read and Compare Phase Reports for the Direct Technique

Reports from the read and compare phase of the direct technique differ, depending on whether or not you specify the READONLY keyword in the CHECK control statement during the initialization phase.

With READONLY:

- The IMS database and the DB2 tables to be checked are all in a read-only status.
- No concurrent update program is running against the data between the time the CCU starts and ends the read phase.
- The read and compare phase creates the repair data set and prints the data mismatches as the data is read and compared.

Without READONLY:

- Concurrent update jobs can run against the data involved in the check process.
- The CCU writes mismatch indication records and requires that the error location phase be run if data inconsistencies occurred and if you did not specify the optional keyword KEYONLY on the CHECK control statement.

The CCU generates the following reports during the read and compare phase of the direct technique:

- Start messages
- Information and warning messages
- Data mismatch messages (with READONLY)
- Segment and row statistics
- End messages

Read and Compare Phase Start Messages

Messages EKYX116I and EKYC300I provide the IMS DPROP system name used when the CCU was submitted, and an initial started message. In Figure 338 on page 480 the IMS DPROP system name is DPROP001.

EKYX116I indicates that the CCU program started reading the IMS DPROP directory master table. The mapping table qualifier of the IMS DPROP system, was specified during DPROPGEN.

```
DPROP120 PGM=EKYC300X JOB/STEP=JOB1 G READ DATE=93.001 TIME=12.00.00  DDN=CCUPRINT PAGE=1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC300I READ AND COMPARE PHASE STARTED
```

Figure 338. The Read and Compare Phase Start Messages

Read and Compare Phase Information and Warning Messages

Most of the information and warning messages issued in the read and compare phase in the direct technique are common to messages issued in the IMS and DB2 read phases of the hashing technique, and to the messages issued in the error location phase.

Refer to “Common Messages for All CCU Phases ” on page 490 for the description of following messages:

```
EKYC312I EKYC815I EKYC816I EKYC818I EKYC820I
EKYC821I EKYC823W EKYC824W EKYC825I EKYC828I
EKYC831W EKYC833I EKYC855I EKYC862W EKYC865W
EKYC906W EKYC935I EKYC936I EKYC956W
```

Message EKYC302I, shown in Figure 339, occurs if you use the DIRECT keyword in the CHECK control statement without the READONLY keyword. It tells you the number of data mismatches written to the //CCUMSMTC data set.

```
EKYC302I 0 MISMATCH INDICATIONS WRITTEN TO THE //CCUMSMTC DATA SET
```

Figure 339. Message EKYC302I

In this example, the CCU did not detect any data mismatches. If there *is* a data mismatch, you must run the error location phase to determine the type of mismatch and which tables and segments are affected. It is possible that a parallel update job was running concurrently with the CCU read phase, falsifying the results of the read and compare phase. If no update job was running in parallel, you can assume that some data really is inconsistent.

Message EKYC303I occurs if you use the READONLY keyword in the CHECK control statement of the initialization phase. Message EKYC303I, tells you the number of detected data mismatches. This is shown in Figure 340.

```
EKYC303I 0 DATA INCONSISTENCIES LISTED WITHIN THIS RUN
```

Figure 340. Message EKYC303I

In this example, the CCU did not detect any data mismatches. If the number of data mismatches is higher than zero, then data was inconsistent when the CCU read it. Message EKYC935I, described on page 500, provides information on the type of mismatch and which tables and segments are affected.

If there were detected mismatches, you can verify whether the IMS and DB2 data was in read-only status during the CCU read phase. Mismatches can be listed and

repair statements generated for data that is updated by a job running in parallel to the CCU, even if there is no real data inconsistency.

Read and Compare Phase Statistics in the Direct Technique

The read and compare phase statistics report tells what data the CCU retrieved from the databases. Figure 341 shows message EKYC310I and EKYC311I, which tell you how many segments and rows were retrieved for specific PRIDs.

```
EKYC310I STATISTICS FOR RETRIEVED IMS AND DB2 DATA:
EKYC311I STATISTICS FOR RETRIEVED IMS AND DB2 DATA AT TIME OF ERROR:

     SEGMENT       RETRIEVED CHECK PRID     SEGMENT ROLE  CONVERTED SEGM                          CCUMSMTC RECORDS
                               MAPPING CASE/OPTIONS   RETRIEVED ROWS DB2 TABLE NAME
     ----------------------- ----- ---------------------- -------------- ------------------------------------------
     S0000                 4 YES   PR0000   ENTITY                    4                                          0
                                   MC1                               4 DB2PROD."T0000"
                             USED  PR5100A  PARENT
                             USED  PR5100B  PARENT
                             ----- ---------------------- -------------- ------------------------------------------
     S1000                 4 YES   PR1000   ENTITY                    4                                          0
                                   MC2                               4 DB2PROD."T1000"
                             ----- ---------------------- -------------- ------------------------------------------
     S1300                 2 YES   PR1000   EXTENSION                 4                                          0
                                   MC2                               - DB2PROD."T1000"
                             ----- ---------------------- -------------- ------------------------------------------
     S2000                 5 NO    N/A
                             ----- ---------------------- -------------- ------------------------------------------
     S3000                 9 YES   PR3000   ENTITY                    9                                          2
                                   MC1                               7 DB2PROD."T3000"
                             ----- ---------------------- -------------- ------------------------------------------
     S4000                 7 YES   PR4000   ENTITY                    3                                          0
                                   MC1 / WHERE                       0 DB2PROD."T4000_VIEW"
                             ----- ---------------------- -------------- ------------------------------------------
     S5000                 4 USED  PR5100A  PARENT
                             USED  PR5100B  PARENT
                             ----- ---------------------- -------------- ------------------------------------------
     S5100                 6 YES   PR5100A  CONTAINING               10                                          1
                                   MC3 / PATHDATA                   10 DB2PROD."T5100A"
                             YES   PR5100B  CONTAINING                5                                          2
                                   MC3 / PATHDATA                    7 DB2PROD."T5100B"
                              ----- ---------------------- -------------- ------------------------------------------
```

Figure 341. The Read and Compare Phase Statistics

The five areas of the report are described in the following list:

Segment

Identifies the segment names that were sensitive in your PSB, and, for each segment type listed, the number of retrieved segments.

If no segments of a specific segment type were retrieved, and you specified your own PSB instead of using the CCU-generated PSB, check whether the segment type with ’0’ occurrences was defined in your PSB. If it was not included in the PSB, add it and resubmit the phase. Refer to Table 40 on page 471 for PSB rules.

CHECK

The CHECK column indicates whether the segment was selected to be checked:

YES The segment type is checked.

NO The segment type is not checked, but it is in the physical path to another segment type that has to be checked. If the segment type

is not in the physical path to another segment that is selected, then retrieving these segments consumes elapsed time.

USED The segment type was used to map a dependent segment type with PATH data or a WHERE clause.

PRID Name and Propagation Request Characteristics

If the segment's role is PARENT, there is one line with the name of the PRID and the role of the segment within this PRID,

If the segment's role is ENTITY, CONTAINING, or EXTENSION, there are two lines. The first one shows the PRID name and the role, and the second one shows some of the propagation request characteristics.

CONVERTED SEGM and RETRIEVED ROWS

The number under CONVERTED SEGM shows you how many of the retrieved IMS segments the RUP converted to the format of a DB2 row.

In Figure 341 on page 481, four segments of type S0000 were retrieved, and the RUP converted all four.

Segment S4000 was retrieved seven times from the IMS database, but the RUP converted only three of them. The reason for this can be either:

- A WHERE clause on this propagation request
- A Segment exit routine that suppresses some of the segments
- Any preceding RUP conversion error, which would be reported in the CCUPRINT data set that precedes this statistic report

Segment S5100 was retrieved six times from the IMS database. For PRID PR5100A, the RUP created ten internal segments and converted them to DB2 format; for PRID PR5100B, the RUP created five internal segments and converted them.

The number under RETRIEVED ROWS shows how many rows of the listed DB2 table were retrieved.

If the program did not retrieve at least one row from a table, determine why no rows were retrieved. For example:

- If you coded a USE keyword, check the definition of the view.
- If the table name is unqualified, the DB2 implicit qualifications might have invalidly qualified the table, and you should determine whether the program accessed the table you wanted to check. Refer to Figure 368 on page 497 for more information on DB2 implicit qualification.

You can now subtract the number of retrieved DB2 rows from the number of converted IMS segments. If the result is 0, this could indicate that every propagated IMS segment has a corresponding DB2 row. If the result is other than 0, then the data might be inconsistent for the PRID.

Segment type S3000, for example, was retrieved nine times from the IMS database, and the RUP converted these nine segments to nine strings that represent the DB2 format of the segment. Only seven rows were retrieved from the DB2 table. Therefore, you can assume that the data for this PRID is not consistent.

Number of CCUMSMTC records.

If this number is 0, then the CCU did not detect data inconsistency within the current run. Any number other than 0 indicates that:

- There were data inconsistencies (see report of segment type S3000 with 2 missing rows, or S5100 with 2 missing internal segments for PRID PR5100B)
- There were data retrieval sequence problems that must be verified in the error location phase (see report of segment type S5100 and PRID PR5100A).

If you are using the direct technique with the READONLY keyword, this column shows MISMATCHES REPORTED instead of CCUMSMTC RECORDS.

Read and Compare Phase End Messages

The CCU read phase ends with one of the messages shown in Figure 342:

```
EKYC304I READ AND COMPARE PHASE ENDED AT END OF DATA
EKYC304I READ AND COMPARE PHASE ENDED AFTER 100 ERRORS

EKYC305I READ AND COMPARE PHASE ENDED WITH ERRORS
```

Figure 342. The Read and Compare Phase End Messages

- EKYC304I indicates that the CCU read phase of the direct technique ended successfully. The data sets, if any, have been created successfully. If you received any warning messages, determine the cause. You can choose to fix them and resubmit the step or not.
- EKYC305I indicates that an error occurred. Scan the read and compare phase reports for error messages and fix the error. The data sets required by subsequent CCU job steps were not created successfully.

Hash Sum Compare Phase Reports for the Hashing Technique

Use the hash sum compare phase of the CCU to compare the hash sum records created by the IMS read phase with those created by the DB2 read phase. Depending on the results of this compare, it sets a first breakpoint in the CCU run; that hash sum report determines whether you must continue with the compare and error location phase.

The hash sum compare phase produces the following reports:
- Start messages
- Compare messages
- End messages

Hash Sum Compare Phase Start Messages

Message EKYC400I, as shown in Figure 343, indicates that the hash sum compare phase has started.

```
DPROP120 PGM=EKYC400X JOB/STEP=JOB1 HSUMCOMP DATE=93.001 TIME=12.00.00 DDN=CCUPRINT PAGE=1

EKYC400I HASH SUM COMPARE PHASE STARTED
```

Figure 343. Message EKYC400I

Hash Sum Compare Messages

In the hashing method, the CCU creates two hash sum records during the IMS read phase, and two hash sum records during the DB2 read phase. At the end of the read phase the CCU writes the sums to //CCUHSUM1 and //CCUHSUM2. For each PRID checked, two hash sum records are created during the IMS read phase, and two hash sum records are created during the DB2 read phase. The first record of a specific PRID of the IMS read phase pairs with the first record of the DB2 read phase for the same PRID, and the second record from the IMS pairs with the second record from the DB2 read phase.

The hash sum compare phase compares each of the pairs and generates message EKYC432I to notify you if the two sums were equal, as shown in Figure 344.

```
EKYC432I HASH SUMS ARE     EQUAL FOR PRID PR0000   SEGMENT S0000    TABLE <DB2PROD."T0000">

EKYC432I HASH SUMS ARE     EQUAL FOR PRID PR1000   SEGMENT S1000    TABLE <DB2PROD."T1000">
```

Figure 344. Hash Sum Compare Messages

The first EKYC432I message in Figure 344 indicates that the data assigned to PRID PR0000 (segment S0000 and the corresponding propagated DB2 table DB2PROD.″T0000″) have the same hash sums. Therefore, you can assume that the IMS segment content matches the row content of the DB2 table.

The CCU continues processing using the hash sums up to the compare phase, after that it uses the actual data.

In the example in Figure 344 PRID PR1000 includes IMS segments S1000 and S1300 of a mapping case 2 propagation request. In the second EKYC432I message, you can see that the hash sums for PRID PR1000 are equal. What you do not see is that PRID PR1000 also includes IMS segment S1300. This is due to the complexity of processing mapping case 2 segments. If you know that PRID PR1000 is a mapping case 2 propagation request, you can assume that not only segment S1000 but also segment S1300 are consistent with the rows of DB2 table DB2PROD.″T1000″.

Figure 345 shows message EKY431W, which you might receive if the data assigned to the PRIDs PR0000 and PR1000 was not consistent when the CCU read the data.

```
EKYC431W HASH SUMS ARE NOT EQUAL FOR PRID PR0000   SEGMENT S0000    TABLE <DB2PROD."T0000">

EKYC431W HASH SUMS ARE NOT EQUAL FOR PRID PR1000   SEGMENT S1000    TABLE <DB2PROD."T1000">
```

Figure 345. Message EKYC431W

Possible reasons for this inconsistency include:

- An update job was running while the CCU was processing the database. The CCU processed a segment or row just as a concurrent job updated it.
- The PSB specified in the IMS read phase doesn't have the segment type identified in message EKYC431W. Scan your IMS read phase reports for message EKYC833I and refer to "Common Messages for All CCU Phases " on page 490 for more information.

- An IMS field or DB2 column identified in message EKYC816I (see page 494) is not yet processed. The note following the message explains how the CCU will handle the data in the field or column in question.

Submit the compare and error location phase for each occurrence of inconsistent data.

In the second message in Figure 345 on page 484, if PRID PR1000 is a mapping case 2 propagation request including IMS segments S1000 and S1300, you will not be able to determine which segment types caused the hash sums to be unequal until you run the compare and error location phase to determine which segment has invalid data.

You receive message EKYC433I, shown in Figure 346, if the CCU retrieved no data for a specific propagation request in either the IMS or DB2 read phases. The CCU cannot compare hash sums for this propagation request.

```
EKYC433I NO HASH SUMS AVAILABLE FOR PRID PR4000   SEGMENT S4000    TABLE <DB2PROD."T4000">
```

Figure 346. Message EKYC433I

Hash Sum Compare Phase End Messages

If all the hash sum pairs from the IMS and DB2 read phases match, you receive message EKYC495I, as shown in Figure 347, and can assume that your data is consistent.

```
EKYC495I HASH SUMS OF DL/I AND DB2 READ PHASES ARE EQUAL
```

Figure 347. Message EKYC495I

The CCU uses hash sums to process data up to the compare phase. After that it uses the real data.

If you received message EKYC431W during the hash sum compare phase, you also receive message EKYC496I, shown in Figure 348. This message summarizes the output of the hash sum compare phase of the current CCU run.

```
EKYC496I HASH SUMS OF DL/I AND DB2 READ PHASES ARE NOT EQUAL
```

Figure 348. Message EKYC496I

EKYC496I does not indicate what data might be inconsistent; it only indicates that there were inconsistencies when the CCU read the databases. To find what data might be inconsistent, look for EKYC431W messages in your report.

The CCU hash sum compare phase ends with one of the messages shown in Figure 349 :

```
EKYC497I HASH SUM COMPARE PHASE ENDED NORMALLY
EKYC498I HASH SUM COMPARE PHASE ENDED WITH WARNINGS

EKYC499I HASH SUM COMPARE PHASE ENDED WITH ERRORS
```

Figure 349. Hash Sum Compare Phase End Messages

- EKYC497I indicates CCU hash sum compare phase has ended successfully. The data sets, if any, were created successfully.
- EKYC498I occurs when the program identifies and prints any warning message. Determine the cause of the warning message. The CCU ended processing successfully.
- EKYC499I occurs if any error occurred. Scan the hash sum compare phase reports for error messages and fix the errors. The data sets required by subsequent CCU job steps were not created successfully.

Compare Phase Reports for the Hashing Technique

If the hash sum compare phase shows that not all hash sums are equal, the CCU reads all the hash records created by the IMS and DB2 read phases. If the propagated data between IMS and DB2 is consistent, then every hash record in //CCUHASH1 should have a paired record in //CCUHASH2. If data mismatches occurred, or if segments or rows are missing, then there will not be a corresponding paired record in //CCUHASH1 or //CCUHASH2. The compare phase reads and sorts all hash records from both files and writes to //CCUMSMTC only those records having no corresponding record.

Message EKYC500I, shown in Figure 350, indicates that the CCU compare phase has started.

```
DPROP120 PGM=EKYC500X JOB/STEP=JOB1 COMPARE DATE=93.001 TIME=12.00.00  DDN=CCUPRINT PAGE=1

EKYC500I COMPARE PHASE STARTED
```

Figure 350. Compare Phase Start Messages

Figure 351 shows message EKYC580I, which indicates how many records the CCU read from //CCUHASH1 and //CCUHASH2. The number of records is the sum of all retrieved IMS segments and all retrieved DB2 rows.

```
EKYC580I 123 RECORDS FROM //CCUHASH1 AND //CCUHASH2 DATA SETS READ
```

Figure 351. Message EKYC580I

Although message EKYC580I indicates how many records have been read, message EKYC581I indicates the number of records *not* written to //CCUMSMTC because they have corresponding records and do not need to be passed to the error location phase. The difference between these two numbers is the number of records whose data content must be verified once again by the CCU error location phase. Figure 352 shows message EKYC580I.

```
EKYC581I 120 RECORDS FROM //CCUHASH1 AND //CCUHASH2 DATA SETS DELETED
```

Figure 352. Message EKYC581I

Message EKYC582I in Figure 353 on page 487 indicates the number of mismatched records. In Figure 353 on page 487 you see there were three mismatched records, which is the number of records in message EKY581I subtracted from the number in message EKYC580I. These three records are passed to the CCU error location phase.

```
EKYC582I 3 MISMATCH RECORDS WRITTEN TO THE //CCUMSMTC DATA SET
```

Figure 353. Message EKYC582I

You receive message EKYC583W, described in Figure 354, if all records that were read have a corresponding record. No error location phase is necessary even though message EKYC583W causes a return code of 4. The reason for this return code is to allow condition code processing in your job control stream, which also allows you to submit the error location phase when necessary. You could not do so without a non-zero condition code.

```
EKYC583W NO RECORDS WRITTEN TO THE //CCUMSMTC DATA SET
```

Figure 354. Message EKYC583W

The CCU compare phase ends with one of the messages shown in Figure 355 :

```
EKYC597I COMPARE PHASE ENDED NORMALLY
EKYC598I COMPARE PHASE ENDED WITH WARNINGS

EKYC599I COMPARE PHASE ENDED WITH ERRORS
```

Figure 355. Compare Phase End Messages

- EKYC597I indicates that the CCU compare phase technique ended successfully and the data sets were created successfully.
- EKYC598I indicates the program generated a warning message. The CCU ends processing successfully.
- EKYC599I indicates an error occurred. Scan the compare phase reports for error messages and fix the error. The data sets required by subsequent CCU job steps were not created successfully.

Error Location Phase Reports

If you used the hashing technique, or if there were concurrent update jobs running when the CCU read the data in the direct technique, then the compare phase or the read and compare phase probably created mismatch records and wrote them to //CCUMSMTC.

The error location phase now reads the mismatch records from //CCUMSMTC. Using these records as an entry point to the data, it reads the corresponding IMS segment and its related DB2 row once again, directly comparing the retrieved data.

The error location phase generates the following reports:
- Start messages
- Information and warning messages
- Data mismatch messages
- Statistics
- End messages

Error Location Phase Start Messages

Message EKYX116I and EKYC600I provide the IMS DPROP system name used when the CCU was submitted, and an initial started message. In Figure 356 the IMS DPROP system name is DPROP001.

```
DPROP120 PGM=EKYC600X JOB/STEP=JOB1 G ERRORLOC DATE=93.001 TIME=12.00.00 DDN=CCUPRINT PAGE=1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC600I ERROR LOCATION PHASE STARTED
```

Figure 356. Error Location Phase Start Messages

Message EKYX116I indicates that the CCU started reading the IMS DPROP directory mapping table. The mapping table qualifier of the IMS DPROP system was specified during DPROPGEN.

Error Location Phase Information and Warning Messages

Most of the information and warning messages issued in the error location phase are common to messages issued in the other phases.

Refer to “Common Messages for All CCU Phases ” on page 490 for the description of following messages:

```
EKYC618I EKYC630X EKYC815I EKYC816I EKYC818I EKYC820I
EKYC821I EKYC825I EKYC828I EKYC831W EKYC833I EKYC855I
EKYC906W EKYC935I EKYC936I EKYC956W
```

Message EKYC610W

In the previous step (the read and compare phase if you used the direct technique, or the compare phase if you used the hashing technique), a data set is created that probably contains data mismatch indication records. The error location phase now reads these records, but either the data set is empty or the named ddname is missing in the JCL. Message EKYC610W is shown in Figure 357.

```
EKYC610W //CCUMSMTC DD STATEMENT MISSING OR DATA SET IS EMPTY
```

Figure 357. Message EKYC610W

Message EKY617I

Scan the output reports from the previous step for message EKYC302I or EKYC583W. If the number of records created by those steps is 0, then you cannot submit the error location phase. If the number of mismatch indication records is other than 0, you see message EKYC617I, shown in Figure 358, which tells you the number of data mismatch indications read.

```
EKYC617I 3 MISMATCH INDICATIONS RECORDS READ FROM //CCUMSMTC DATA SET
```

Figure 358. Message EKYC617I

Message EKYC619I

You might also receive message EKYC619I, telling you that some mismatch indication records were skipped because the program could not detect a real data inconsistency. A possible reason for skipping mismatch indication records is that a parallel update job was running just when the read phases

retrieved the data, and the error location now finds the data consistent again.

```
EKYC619I 2 MISMATCH INDICATIONS RECORDS SKIPPED
```

Figure 359. Message EKYC619I

Message EKYC620I

Message EKYC620I indicates there is a data mismatch between the IMS and DB2 data. The following steps show how the CCU determines that there is a mismatch:

1. The CCU writes a record to //CCUMSMTC during the compare phase.
2. The CCU reads this record and analyzes it during the error location phase.
3. The record is either:
   - A DB2 row that is missing the change that exists in the corresponding IMS segment, if the propagation request mapping direction is hierarchical to relational.
   - An IMS segment that does not reflect the change shown in the corresponding DB2 row, if the propagation request mapping direction is relational to hierarchical.
4. Assume the record was identified as an IMS segment that is missing the corresponding update shown in a DB2 row.

   In earlier phases of the CCU job, the program indicates that some problems occurred while retrieving the data; for example, a segment or row was not available or a conversion error occurred when the data was accessed. The CCU generates a mismatch indication record and writes it to //CCUMSMTC to let the error location phase verify that segment or row once again.

   The error location phase program reads the record. Because the error is in an IMS segment, the CCU issues qualified GU (get unique) calls.
5. After retrieving the required segment, the relational update program (RUP) (RUP) is called to convert the data to get the values for the corresponding propagated DB2 row.

   If it is a DB2 primary key that originates from the newly converted IMS segment key, a qualified SELECT statement is issued to retrieve the DB2 row.
6. After that, the CCU compares the data between the IMS segment and the DB2 row.

   However, if the mapping direction is hierarchical to relational, and if IMS DPROP does not support the conversion of a DB2 row to the value and format of an IMS segment, then the only thing that the CCU can do is issue message EKYC620I and print the DB2 primary key. You must verify whether the data inconsistency actually exists.

   If the propagation request is hierarchical to relational, but the mismatch record that was read from //CCUMSMTC points to a DB2 row instead, then the CCU should process in that direction properly.

Figure 360 on page 490 shows two examples of message EKYC620I.

```
EKYC620I PROBABLE MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."TD0000">
         DB2 ROW KEY            +0000   F8F9F9F1 F1F1                 *899111          *
         PROGRAM CANNOT TRY TO RETRIEVE THE CORRESPONDING SEGMENT FROM THE DL/I DATABASE

EKYC620I PROBABLE MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."TD0000">
         SEGMENT KEY            +0000   F8F9F9F1 F1F1                 *899112          *
         PROGRAM CANNOT TRY TO RETRIEVE THE CORRESPONDING ROW FROM THE DB2 TABLE
```

Figure 360. Probable Mismatching Data

Error Location Phase Statistics

```
EKYC614I DATA MISMATCH STATISTICS:

         PRID     SEGMENT  DB2 TABLE NAME                    REPORTED MISMATCHES
         -------- -------- --------------------------------- -------------------
         PR0000   S0000    DB2PROD."T0000"                                     0
         PR1000   S1000    DB2PROD."T1000"                                     0
         PR3000   S3000    DB2PROD."T3000"                                     0
         PR5100A  S5100A   DB2PROD."T5100A"                                    0
         PR5100B  S5100B   DB2PROD."T5100B"                                    1
```

Figure 361. Error Location Phase Statistics

Message EKYC614I in Figure 361 indicates the number of data inconsistencies for each PRID. In Figure 361 there was one data inconsistency for PR5100B. The EKYC935I message preceding this mismatch statistic shows the type of inconsistency, and in which segment and row the inconsistency occurred.

Error Location Phase End Messages

The CCU error location phase ends with one of the messages shown in Figure 362:

```
EKYC602I ERROR LOCATION PHASE ENDED AT END OF DATA

EKYC603I ERROR LOCATION PHASE ENDED WITH ERRORS
```

Figure 362. Error Location Phase End Messages

- EKYC602I indicates that the CCU error location phase has completed successfully, and all data from the //CCUMSMTC data set was processed. The repair data sets //CCUIERR and //CCUDERR were created successfully. If you received any warning messages, determine what caused the warning, fix it, and resubmit the step. The CCU ended processing successfully.
- EKYC603I is printed if any error situation occurs. Scan the error location phase reports for error messages, fix the error situation, and resubmit the job.

Common Messages for All CCU Phases

Table 41 describes common CCU messages, what CCU phase issues them, and what page to go to for the message description. Most of the messages are common to the following phases:

- IMS and DB2 read phase of the hashing method
- Read and compare phase of the direct method
- Error location phase

Table 41. Common Messages

Message	Issued by Phase	Description on Page
EKYC312I	R	491

Table 41. Common Messages (continued)

Message	Issued by Phase	Description on Page
EKYC618I	E	492
EKYC630X	E	492
EKYC815I	I, D, R, E	493
EKYC816I	R, E	494
EKYC818I	D, R, E	496
EKYC820I	D, R, E	497
EKYC821I	I, D, R, E	497
EKYC823W	R, E	498
EKYC824W	D, R, E	498
EKYC825I	I, D, R, E	498
EKYC828I	I, D, R, E	499
EKYC831W	I, R, E	499
EKYC833I	I, R, E	500
EKYC855I	D, R, E	498
EKYC862W	I, D, R	493
EKYC865W	I, D, R	492
EKYC906W	I, R, E	500
EKYC935I	R, E	500
EKYC936I	R, E	503
EKYC956W	R, E	503

Note:

Phase
I = IMS Read Phase of the Hashing Technique
D = DB2 Read Phase of the Hashing Technique
R = Read and Compare Phase of the Direct Technique
E = Error Location Phase

EKYC312I and EKYC618I Using the FORCE Keyword

You can use the optional FORCE keyword of the CHECK control statement to use the direct technique for an IMS database that is normally not supported in the direct technique. See the*Administrators Guide* for rules covering direct technique.

If you use the FORCE keyword, all the fully concatenated keys of all IMS segments to be checked must be retrieved in the same key sequence as the DB2 primary keys of the corresponding DB2 tables. The CCU issues message EKYC312I or EKYC618I, shown in Figure 363 on page 492, when data mismatches that involve the FORCE keyword occur.

```
EKYC312I CCU DATA MISMATCH INDICATIONS CAN BE CAUSED BY SPECIFYING THE FORCE KEYWORD IN THE CHECK INPUT COMMAND

EKYC618I CCU DATA MISMATCH INDICATIONS CAN BE CAUSED BY SPECIFYING THE FORCE KEYWORD IN THE CHECK INPUT COMMAND
```

Figure 363. Message EKYC312I and EKYC618I

Scenario

Here is a scenario where you might FORCE the DIRECT keyword:

You are using an HDAM database from which you access the IMS root segments with unqualified get next (GN) calls. The root segments are not retrieved in any particular key sequence, but in the sequence your randomizing module stored them. You can write your own HDAM randomizing module to put the IMS segments in a key sequence that keeps the fully concatenated keys in ascending order. Then you can use the FORCE keyword to force the CCU to use the DIRECT technique instead of the hashing technique for that database.

Scenario

The following scenario illustrates problems that can occur if you FORCE the CCU to use the direct technique:

You are using an HDAM database and the IMS root segment retrieve sequence does not match the retrieve sequence of the DB2 row. The contents of the database and the corresponding table are:

IMS Segment	DB2 Row
A14	A10
A16	A12
A12	A14
A10	A16

With the direct technique, the CCU sequentially accesses the IMS database segments (A14, A16, A12, A10), and compares them with their corresponding rows. Note that the IMS segments are in random order, and the DB2 rows are in ascending order. When the CCU compares segment A14 with row A10, and so on, it will generate a report for each mismatch.

This compare can lead to an immense error list, however, most of the printed data mismatches are presumably not mismatches. They just appear to be errors because the wrong CCU technique was used. This example is true for all databases where the retrieve sequence of the fully concatenated segment key does not match the retrieve sequence of the DB2 primary key of the corresponding DB2 table. Refer to “Hashing and Direct Techniques ” on page 422 for more information on this subject.

EKYC630W, EKYC862W and EKYC865W

Figure 364 on page 493 shows messages EKYC630W, EKYC862W and EKYC865W. The CCU issues these messages when processing stops because the error limit was reached. You specify the error limit on the MAXERROR keyword of the CHECK control statement.

```
EKYC630W MAXIMUM ERROR LIMIT 100 REACHED. PROGRAM IS TERMINATED.

EKYC862W MAXIMUM ERROR LIMIT 100 REACHED. FORCED PROGRAM TERMINATION.

EKYC865W MAXIMUM ERROR LIMIT 100 REACHED. PROGRAM IS TERMINATED.
```

Figure 364. Message EKYC630W, EKYC862W and EKYC865W

The forced termination of the CCU can happen in the following circumstances:

1. You use the direct technique and the READONLY keyword on the CHECK control statement. The CCU detects the number of data mismatches specified on the MAXERR keyword of the CHECK control statement and stops processing. The repair data sets //CCUIERR and //CCUDERR have been created successfully and you can use them to repair the data mismatches. Then resubmit the CCU job with the MAXERROR keyword specifying a higher number than the one you used before, because more data mismatches could exist for this database.
2. You use the direct technique but do not specify the READONLY keyword on the CHECK control statement, the CCU writes the number of data mismatch indication records to //CCUMSMTC and the CCU stops processing. You can pass //CCUMSMTC to the error location phase to verify the mismatch indications. Resubmit the CCU job, specifying MAXERROR and a number higher than the number you previously specified, because more data mismatches could exist for this database.
3. The CCU calls the RUP to convert the data for a retrieved IMS segment, but the RUP cannot convert the data. Instead, it writes an error message to //CCUPRINT and returns a nonzero return code to the CCU. The CCU continues processing, but the segment that caused the data conversion error in the RUP is not processed. After the limit of errors, as specified on the MAXERROR keyword of the CHECK control statement, has occurred, the CCU stops processing. Before resubmitting the job with a higher value in MAXERROR, verify the RUP error messages.
4. You use unqualified DB2 tables. When you use unqualified tables, DB2 implicit table qualification is in effect, and DB2 allocates an application table that should not be checked. Scan your output list for message EKYC820I and refer to Figure 368 on page 497.
5. You use DB2 views for the tables that should be checked. Scan the CCU output list for message EKYC818I and refer to Figure 367 on page 496. If EKYC818I messages were written to //CCUPRINT, you can verify the data mismatch messages with the DB2 tables that you want to check.

EKYC815I

Information message EKYC815I indicates that the CCU has read the control data set that was created during the CCU initialization phase; it includes the processing options and the timestamp. Figure 365 shows an example of message EKYC815I.

```
EKYC815I CONTROL DATA SET SPECIFICATIONS:
         TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING HASHING TECHNIQUE.
         DPROP SYSTEM NAME=DB2PROD  TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS
```

Figure 365. Message EKYC815I

The timestamp must be the same in all the CCU phases within one CCU run. See “Control Data Set Timestamp ” on page 474 for more information on control data set timestamps.

EKYC816I

Figure 366 shows an example of message EKYC816I. This message indicates that the CCU will perform special mapping for the fields and columns listed in the message.

```
EKYC816I THE FOLLOWING IMS FIELD AND DB2 COLUMN DEFINITIONS MIGHT CAUSE DATA COMPARE PROBLEMS IN THE CCU.
         FOR MORE INFORMATION, SEE 'NOTES' FOLLOWING THIS LIST.

       PRID     SEGMENT  FIELD    DATATYPE DB2 COLUMN NAME    DATATYPE NULLS  DB2 TABLE NAME                      NOTE
      -------- -------- -------- -------- ------------------ -------- ------ ----------------------------------- ----
      PR3000   S3000    F3000IN1 B        C3000IN1           SMALLINT YES    DB2PROD."T3000"                       2
                        F3000IN2 B        C3000IN2           INTEGER  YES                                          2
                        F3000IN3 B        C3000DC1           DECIMAL  YES                                          2
                        F3000PK  PACKED   C3000DC2           DECIMAL  W.DEF.                                       5
                        F3000TMS TIMESTMP C3000TMS           TIMESTMP W.DEF.                                       3
      PR5100A  S5100A   F5100LVG LONGVARG C5100VG            VARG     W.DEF. DB2PROD."T5100A"                      1
      PR5100B  S5100B   F5100TIM TIME     C5100TIM           TIME     W.DEF. DB2PROD."T5100B"                      4
                        F5100FLT FLOAT(E) C5100FLT           FLOAT(D) W.DEF. DB2PROD."T5100B"                      6

      NOTES:  1) TRAILING BLANK CHARACTERS (X'40') OF THE FIELD AND COLUMN DATA WILL BE TRUNCATED BEFORE COMPARISON.
              2) BLANK OR ZERO FIELD DATA COMPARED WITH DB2 COLUMN 'NULLS' WILL NOT BE CONSIDERED AS DATA MISMATCH.
              3) TIMESTAMPS HAVE DIFFERENT LENGTH; ONLY THE COMMON TIMESTAMP DATA VALUE WILL BE USED FOR COMPARISON.
              4) COLUMN DEFINITION 'NOT NULL WITH DEFAULT'. FIELD WILL ONLY BE CHECKED IF AVAILABLE IN IMS SEGMENT.
              5) SCALE FACTORS ARE DIFFERENT; ONLY THE COMMON PART OF THE SCALE VALUE WILL BE USED FOR COMPARISON.
              6) FIELD OR COLUMN WITH DOUBLE PRECISION FLOATING POINT DATA IS TRUNCATED OR ROUNDED TO SINGLE FLOAT.
```

Figure 366. Message EKYC816I

The last column identifies a note that describes how the CCU processes the fields or columns. The notes follow the text of the message; they are:

Note 1)

TRAILING BLANK CHARACTERS (X'40') OF THE FIELD AND COLUMN DATA WILL BE TRUNCATED BEFORE COMPARISON

In Figure 366, field F5100LVG has a data type of LONGVARG, and the corresponding DB2 column C5100VG has a data type of VARG.

For any of the fields or columns having a variable-length data type, the program truncates all trailing blanks of the field *and* column before it compares the data values. This includes data types VARCHAR, LONG VARCHAR, VARG and LONGVARG, and is also true if only one of the fields or columns is variable length.

Note 2)

BLANK OR ZERO FIELD DATA COMPARED WITH DB2 COLUMN ’NULLS’ WILL NOT BE CONSIDERED AS DATA MISMATCH

In DB2, you can define a DB2 column as nullable. When you insert a DB2 row and do not specify a value for a nullable column, DB2 assigns NULLS to it. In IMS, however, if you insert a segment and do not provide a value for a field, IMS assigns to the length field the initial value as specified at DBDGEN, or simply uses what it finds in your program buffer.

The CCU does not report data inconsistencies if the DB2 column is NULLS and the IMS field value contains all blanks or zeroes only.

Note 3)

TIMESTAMPS HAVE DIFFERENT LENGTH; ONLY THE COMMON TIMESTAMP DATA VALUE WILL BE USED FOR COMPARISON

You receive this note if you have declared an IMS field to IMS DPROP with a TIMESTAMP data type having a length that is shorter than the corresponding propagated DB2 column TIMESTAMP.

The CCU truncates the longer timestamp value to the length of the shorter timestamp before comparing any data. In the example in Figure 366 on page 494, the CCU truncates the timestamp of the DB2 column to 10 bytes before the resulting value is compared with the 10 bytes that come from the IMS field's timestamp.

Note 4)

COLUMN DEFINITION 'NOT NULL WITH DEFAULT'. FIELD WILL ONLY BE CHECKED IF AVAILABLE IN IMS SEGMENT

The CCU cannot compare a DB2 column having a data type of DATE, TIME or TIME STAMP defined with NOT NULLS WITH DEFAULT, with the data content of the IMS field. This is because DB2 can set the value for the DATE, TIME or TIMESTAMP column when you insert the DB2 row.

Here is an example:

You have a mapping case 1 propagation request. Segment S1 is the entity, and T1 is the name of the DB2 table. Segment S1 is defined as variable length, with a minimum length of 50 bytes and a maximum length of 100 bytes. Further assume that the segment contains a field FTIME, whose length is 8 bytes, starting in position 90. Table T1 has a column, CTIME, which was defined with NOT NULL WITH DEFAULT.

If an application program inserts segment S1 with a length of 60 bytes at 11:20:05 AM, the data for field FTIME will not be available in the segment; however, in the DB2 row, due to the NOT NULL WITH DEFAULT definition, the column contains the value 11:20:05.

The CCU reads the IMS segment and passes it to the RUP to convert the IMS fields to the format of the corresponding DB2 columns. The CCU processes this S1 segment at 08:00:00 AM, then the RUP sets a value of 08:00:00 for the "missing" field FTIME, which does not correspond to the value 11:20:05 returned from the DB2 column CTIME.

Therefore, the CCU does not check this data if the DB2 column has data type DATE, TIME and a definition of NOT NULLS WITH DEFAULT.

Note 5)

SCALE FACTORS ARE DIFFERENT; ONLY THE COMMON PART OF THE SCALE VALUE WILL BE USED FOR COMPARISON

With IMS DPROP, it is possible to define numeric data types having different scale factors between IMS fields and DB2 columns. Because of this, during regular propagation, the RUP either:

- Truncates the IMS field scale value, or
- Adds trailing zeroes, when inserting a DB2 row.

The HUP either:

- Truncates the DB2 column scale value, or
- Adds trailing zeroes, when inserting an IMS segment.

Before comparing such fields and columns, the CCU truncates the value of the longer scale factor to the length of the shorter scale factor.

Note 6)

FIELD OR COLUMN WITH DOUBLE PRECISION FLOATING POINT DATA IS TRUNCATED OR ROUNDED TO SINGLE FLOAT

Comparing single precision floating point data with double precision floating point data causes presumably unneeded mismatch indications in the CCU.

While processing a DB2 column with double precision floating point data, the program rounds and truncates the double precision floating point data to single precision floating point data. Then, it compares the single precision floating point data from the IMS field with the truncated and rounded double precision floating point data of the DB2 column.

EKYC818I

The CCU can use the USE keyword of the CHECK control statement to specify that the CCU will use a DB2 view instead of a DB2 table name to access data during the read phase. With the USE keyword, the CCU generates an SQL statement for the DB2 table in the initialization phase and replaces the DB2 table name with the view name you specified. Message EKYC8818I, shown in Figure 367, indicates the tables for which you specified a view name.

```
EKYC818I DATA MISMATCH INDICATIONS BETWEEN DL/I AND DB2 DATA MIGHT NOT BE REAL DATA INCONSISTENCIES
         BECAUSE A DB2 VIEW NAME WAS SPECIFIED TO BE USED FOR THE FOLLOWING TABLES

         PRID   DB2 TABLE NAME AS IS IN THE DPROP DIRECTORY IS OVERWRITTEN BY THE FOLLOWING DB2 OBJECT NAME
         ------ ------------------------------------------- ---------------------------------------------
         PR3000 DB2PROD.T3000                               DB2PROD.T3000VIEW
```

Figure 367. Message EKYC818I

The following example describes the use of the view name:

DB2 table DB2PROD.″T3000″ is defined. It contains the propagated data from IMS database DB1. An alternate table, USER.TABLE1, contains rows necessary for your application.

Propagation of IMS data from database DB1 to table DB2PROD.″T3000″ invokes a segment exit routine. Assume that you want to suppress propagation for all segments in DB1 having an ‘A’ in their key value. When you issue a DL/I delete call against the segments in database DB1, the segment exit routine prevents deletion of the DB2 rows from DB2PROD.″T3000″ if the corresponding segment has an ‘A’ in its key value.

If you delete 10 segments that have an ‘A’ in their key value from database DB1, you still have 10 rows in DB2PROD.″T3000″ that are not deleted. The CCU reports data mismatches for those 10 rows unless you use a DB2 view to exclude all rows with an ‘A’ in their key.

DB2PROD.″T3000VIEW″ is a view definition for table USER.TABLE1. When the CCU checks database DB1, its CHECK control statement includes the keyword USE=(DB2PROD.″T3000″,DB2PROD.″T3000VIEW″). The CCU, therefore, does not read the data from DB2PROD.″T3000″ but from USER.TABLE1. In this case, the CCU generates an immense data mismatch list because an incorrect view name was specified in the CHECK control statement.

If the CCU uses a view to retrieve data, it does not check whether:

- The view exists
- All columns defined to the view also exist in the table
- All columns that are propagated were defined for the view when it was defined
- The view definition contains a WHERE clause, and what the WHERE clause includes or suppresses
- The view was defined for the DB2 table that is subject to data propagation

EKYC820I

The CCU generates dynamic SQL statements for the application tables to be checked. If the IMS DPROP mapping tables do not contain a DB2 qualifier for a DB2 table that must be checked, you are notified that DB2 implicit table qualification is in effect. Message EKYC820I, in Figure 368, indicates the DB2 tables for which DB2 uses implicit qualification.

```
EKYC820I JOB WILL USE DB2 IMPLICIT TABLE QUALIFICATION FOR THE FOLLOWING DB2 TABLES
         BECAUSE THEY HAVE NO QUALIFIER IN THE DPROP DIRECTORY AND NO USE= OR QUALIFIER=
         KEYWORD WAS SPECIFIED

         PRID     DB2 TABLE NAME
         -------- --------------------------------------------------
         PR4000   T4000
```

Figure 368. Implicit Table Qualification Message

If data mismatches are detected in subsequent CCU job steps, the generated repair statements for the DB2 tables have no real DB2 table qualifier. Instead, the repair statements are created with eight leading asterisks (*). Because you must replace the eight asterisks by the real DB2 table qualifier before submitting the job, you are prevented from submitting a repair job to update tables you probably do not want to update.

EKYC821I

Message EKYC821I, in Figure 369, warns you that the CCU might encounter data mapping problems for the PRIDs listed after the message.

```
EKYC821I DATA MAPPING PROBLEMS CAN ARISE BETWEEN IMS FIELDS AND DB2 COLUMNS WITH DIFFERENT
         SCALE FACTORS, IF THE FIELD AND/OR COLUMN BELONGS TO THE MAPPED DB2 PRIMARY KEY

         PRID    SEGMENT  FIELD  KEY DEFINITION    COLUMN   KEY DEFINITION   DB2 TABLE NAME
         ------- -------- ------ --- ------------- -------- --- ------------ ---------------
         PR5000  S5000    K5000  YES DECIMAL(5,3)  C5000    YES DECIMAL(6,3) DB2PROD.T5000
```

Figure 369. Message EKYC821I

Scenario

This scenario illustrates an example of a data mismatch.

Assume you have a segment S5000 with the key field K5000. K5000 is defined as DECIMAL(5,3), 5 numeric digits, 3 of which are used to build the scale. The propagated DB2 table T5000 has a key column C5000 with the definition DECIMAL(6,4). Therefore, the DB2 column has one more scale digit than the IMS field.

Assume you insert a new row with the value 99,1234 into T5000. The table is propagated, and HUP builds the IMS segment S5000. Because the IMS field K5000 is DECIMAL(5,3), the HUP truncates the column data to become 99,123. The CCU, when submitted to compare the data, reads this segment and passes it to the RUP for data conversion. RUP builds, based on the DB2 column definition for this field, a value of 99,1230, which is not the same compared with the value 99,1234 returned by DB2 when column C5000 is read. In this example, the CCU creates a DB2 INSERT and IMS DLET repair statement for the key 99,1230, and a DB2 DELETE and IMS ISRT statement for the key 99,1234.

CCU errors can result when numeric fields are propagated in this way. There is no CCU-supported solution at this time. You can write your own program to check the consistency of such propagation requests, or you can disallow generation of propagation requests with such mapping definitions. If you can ensure that rows with values such as 99,123 are not inserted, but rows with values such as 99,123 are inserted, then the CCU can compare the data successfully.

EKYC823W

Figure 370 illustrates message EKYC823W. For a description of this message, see message EKYC097W in Figure 329 on page 474.

```
EKYC823W COMBINING THE CCU DIRECT TECHNIQUE WITH A DESCENDING ORDERING SEQUENCE OF DB2
         PRIMARY KEY COLUMNS MIGHT LEAD TO GENERATED REPAIR FILES THAT SHOULD BE CAREFULLY
         ANALYZED BEFORE APPLYING THEM TO THE IMS AND/OR DB2 DATA

         PRID     SEGMENT  FIELD    DB2 TABLE                COLUMN             ORDERING
         -------- -------- -------- ------------------------ ------------------ --------
         PR2000   S2000    F2000    DB2PROD.T2000            C2000                  DESC
```

Figure 370. Message EKYC823W

EKYC824W

Figure 371 illustrates message EKYC824W. For a description of this message, see message EKYC069W in Figure 323 on page 470.

```
EKYC824W DATA MISMATCH INDICATIONS CAN BE CAUSED BY COMBINING THE DIRECT TECHNIQUE
         WITH THE KEY MAPPING OF A NUMERIC IMS FIELD TO A DB2 PRIMARY KEY COLUMN HAVING A
         NUMERIC DATATYPE, IF THE VALUES OF THESE FIELDS/COLUMNS CAN CONTAIN BOTH POSITIVE
         AND NEGATIVE NUMBERS

         PRID     SEGMENT  FIELD    DATATYPE DB2 COLUMN NAME    COLTYPE  DB2 TABLE NAME
         -------- -------- -------- -------- ------------------ -------- ------------------
         PR2000   S2000    K2000    DECIMAL  C2000              INTEGER  DB2PROD.T2000
```

Figure 371. Message EKYC824W

EKYC825I and EKYC855I

Messages EKYC825I and EKYC855I always occur after DB2 returns an unacceptable SQLCODE.

```
EKYC850E NONZERO SQLCODE OPENING DB2 CURSOR FOR A DYNAMIC SELECT STMT ON TABLE <DB2PROD."T5100A"> FOR PRID PR5100A

EKYZ360E DSNT408I SQLCODE = -204, ERROR:  DB2PROD."T5100A" IS AN UNDEFINED NAME
         DSNT415I SQLERRP = DSNXERT SQL PROCEDURE DETECTING ERROR
         DSNT416I SQLERRD = 210  0  0  1  0  0 SQL DIAGNOSTIC INFORMATION
         DSNT416I SQLERRD = X'FFFFFF2E'  X'00000000'  X'00000000'  X'FFFFFFFF'  X'00000000'  X'00000000' SQL
                  DIAGNOSTIC INFORMATION

EKYC825I THE FOLLOWING SQL STATEMENT CAUSED ABOVE MESSAGE:
         SELECT "K0000", "K5000", "C5000", "C5001", "C5002", "K5100",
            "C5100VG" FROM DB2PROD."T5100A"
```

Figure 372. Message EKYC825I

As shown in Figure 372, messages EKYC850E and EKYZ360E appear together with EKYC825I, to describe one of the following errors:

- EKYC850E indicates that the program issued an OPEN CURSOR instruction for the named DB2 table, and DB2 returned an SQLCODE other than zero.
- EKYZ360E is issued if a bad SQLCODE was returned from DB2. In this case, the CCU passes the SQLCA to the DSNTIAR module, which translates the

SQLCODE to the text that appears following message EKYZ360E. In the example, the OPEN CURSOR instruction was terminated because DB2 could not find the DB2 table.

- EKYC825I gives the SQL statement used when the error occurred.

  Depending on the error, you can use this SQL statement to query the DB2 catalog or directory to determine and fix the error.

EKYC828I

Figure 373 illustrates message EKYC828I, which lists the PRIDs that can cause data mismatch errors due to non-active status or having specified ERROPT=IGNORE.

```
EKYC828I THE DPROP DIRECTORY INDICATES ERROPT(PROPAGATION FAILURE)=IGNORE
         OR A STATUS OTHER THAN 'ACTIVE' FOR THE FOLLOWING PRIDS:

         PRID     STATUS    ERROPT  SEGMENT  DB2 TABLE NAME
         -------- --------- ------- -------- ------------------------------------
         PR0000   ACTIVE    IGNORE  S0000    DB2PROD."T0000"
         PR5100A  INACTIVE  BACKOUT S5100A   DB2PROD."T5100A"
         PR5100B  SUSPENDED IGNORE  S5100B   DB2PROD."T5100B"
```

Figure 373. Message EKYC828I

When you submit the CCU to check data consistency, it accesses the IMS DPROP directory to determine the status of the PRIDs to be checked.

The PRIDs that are shown following the message, either have ERROPT=IGNORE, meaning ignore propagation failures rather than perform a rollback call for the data; or have a status other than ACTIVE, which is not normal for propagation.

If you receive any data mismatch indications for the listed PRIDs, this message identifies one of the following reasons for a data mismatch:

- The RUP or the HUP could not map an IMS segment or a DB2 row because with ERROPT=IGNORE, an update to either one will not be propagated.
- With a propagation request status of SUSPENDED or INACTIVE, there is probably nonpropagated data in either IMS or DB2.

EKYC831W

Message EKYC831W indicates the CCU detected that the parameter list passed from DFSRRC00 to the CCU contains more than two addresses. The CCU considers this a warning. EKYC831I is illustrated in Figure 374.

```
EKYC831W MORE THAN TWO PARAMETERS RECEIVED FROM DFSRRC00:
         - USING 1ST ADDRESS AS I/O PCB AND 2ND ADDRESS AS DBPCB
         - IGNORING ALL OTHERS
```

Figure 374. Message EKYC831W

Possible reasons for the warning shown in Figure 374 include:

- The first parameter points to an I/O PCB, and the second points to a DB PCB containing all required segments of the database to be checked. The CCU works as expected. The CCU continues to work with the I/O PCB and uses the first DB PCB to retrieve segments. The CCU ignores all other addresses.

- If the first parameter points to something other than an I/O PCB, IMS rejects subsequent DL/I INQY calls. The CCU prints an error message and ends processing.
- If the second parameter points to something other than the required DB PCB, subsequent DL/I GET NEXT calls might retrieve segments, but presumably not from the intended IMS database. The CCU then prints an error message and ends processing.

For more information refer to “PSB Concerns for the CCU” on page 471.

EKYC833I

You receive EKYC833I, as shown in Figure 375, if you specified in the JCL for the IMS read phase a PSB name (USERPSB, in this message) other than the PSB name generated by the CCU (EKYCCU00).

```
EKYC833I NAME OF CCU GENERATED PSB IS EKYCCU00, ACTUALLY USING A PSB WITH NAME USERPSB
```

Figure 375. Message EKYC833I

For more information on user-generated PSBs, refer to “PSB Concerns for the CCU” on page 471.

EKYC906W

Message EKYC906W indicates the RUP detected a mapping error for the extension segment’s parent segment, for mapping case 2 extension segments.

Figure 376 illustrates Message EKYC906W.

```
EKYC906W SEGMENT S1300 WITH ROLE=EXTENSION IN PRID PR1000 CANNOT BE PROCESSED
         BECAUSE RUP REPORTED A CONVERSION ERROR FOR ITS ENTITY SEGMENT. SEGMENT SKIPPED
```

Figure 376. Message EKYC906W

In Figure 376, one mapping case 2 entity and one extension segment build the data for one propagated row in a DB2 table. If the RUP detects a mapping error on the entity segment, the CCU is not able to process the entity segment or the extension segment of this mapping case 2 propagation request, and it skips both the entity segment and the extension segment. The CCU generates an IMS repair data set that includes one DL/I ISRT statement for the entity and one for the extension segment. It generates a DB2 repair data set containing a DB2 DELETE statement for the row.

Data Mismatch Reports

During the error location phase, the CCU generates a report for every data mismatch and creates repair statements you can use to fix the data inconsistency. The CCU also generates this report when you use the direct technique and specify the READONLY keyword on the CHECK control statement and the CCU identifies a data mismatch between the IMS database and the related propagated DB2 table.

See “The Generated Repair Data Sets ” on page 504 for a discussion of the layout of the IMS and DB2 repair statements generated by the CCU.

The CCU can generate four different types of reports:

- Probable mismatching data, shown in Figure 360 on page 490, occurs during the error location phase only.
- DB2 row missing, shown in Figure 377, occurs when an IMS segment is retrieved from the database, but the corresponding DB2 row was not located.
- IMS segment missing, shown in Figure 378 on page 502, occurs when a DB2 row is retrieved from the table, but the corresponding IMS segment is not in the database.
- Mismatching data, shown in Figure 379 on page 503, occurs if the segment and the row are found but the propagated “non-key” data had different values.

DB2 Row is Missing

In the example in Figure 377, the CCU retrieved an IMS segment and tried to retrieve a row with the same key from the related DB2 table. DB2 returned an SQLCODE +100 that generated the message EKYC935I.

The CCU considers this ROW NOT FOUND, and it:
- Generates an SQL INSERT statement for the row.
- Generates a DL/I DLET statement for the segment.
- Writes the statements to the repair data sets.
- Generates message EKYC935I.

```
EKYC935I MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."T5100">

         IMS KEY/ID FIELDS TO SEGMENT S5100
          KEY FIELD K0000  IN SEGMENT S0000   +0000   F2F2F1F6 F0F5        *221605       *
          KEY FIELD K5000  IN SEGMENT S5000   +0000   F0F5                 *05           *
          ID  FIELD F5000  IN SEGMENT S5000   +0000   F0F5                 *05           *
          ID  FIELD F5001  IN SEGMENT S5000   +0000   F0F540               *05           *
          ID  FIELD F5002  IN SEGMENT S5000   +0000   00000000 0000500C    *......&.     *
          KEY FIELD K5100  IN SEGMENT S5100   +0000   F0F5                 *05           *

         WITHOUT CORRESPONDING ROW IN TABLE

         ** GENERATED REPAIR STATEMENT NUMBER:              1 **
```

Figure 377. DB2 Row is Missing

Message EKYC935I provides the following information:
- The name of the DB2 table or view for which the error was detected.
- The segment name and the IMS segment key, in a 16-byte dump format. The key consists of:
  - The value of the DB PCB key feedback area.
  - Additional SSA fields (ID fields) declared to IMS DPROP to make the IMS key unique.

    If the propagation request definition is such that an SSA field makes the IMS segment unique, the message contains additional lines with the name and the value of the ID fields.

  The printed IMS key must differ from the value of the corresponding DB2 primary key, because key conversion routines in the RUP or in segment or field exit routines can modify the key values between the segment and the row.
- A line indicating that a row is missing.
- A line identifying the generated repair statement number. This number corresponds to the repair statement in the //CCUDERR or //CCUIERR repair files, or both.

IMS Segment is Missing

In the example in Figure 378, the CCU retrieved a DB2 row but found that no IMS segment with the same key exists in the database. The CCU considers this SEGMENT NOT FOUND and it:

- Generates an SQL DELETE statement for the row.
- Generates a DL/I ISRT statement for the segment.
- Writes these statements to the repair data sets.
- Prints message EKYC935I.

```
EKYC935I MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."T5100">

         DB2 PRIMARY KEY
          KEY COLUMN <K0000>    +0000   F2F2F1F6 F0F5         *221605       *
          KEY COLUMN <K5000>    +0000   F0F5                  *05           *
          KEY COLUMN <K5001>    +0000   F0F5                  *05           *
          KEY COLUMN <K5002>    +0000   F0F540                *05           *
          KEY COLUMN <K5003>    +0000   00000000 0000500C     *......&.     *
          KEY COLUMN <K5100>    +0000   F0F5                  *05           *

         WITHOUT CORRESPONDING IMS SEGMENT S5100 IN THE DATABASE

         ** GENERATED REPAIR STATEMENT NUMBER:              2 **
```

Figure 378. IMS Segment is Missing

Message EKYC935I also provides the following information:

- The name of the DB2 table or view for which the error was detected.
- The DB2 row key in a 16-byte dump format. The key is the DB2 primary key of the table, not the key of the missing IMS segment. If you want the key of the corresponding IMS segment, you must manually convert the printed DB2 key to a value defined for the IMS segment.
- A line indicating that a segment is missing.
- A line identifying the generated repair statement number. This number corresponds to the repair statement in the //CCUDERR or //CCUIERR repair files, or both.

Mismatching Data

In the example in Figure 379 on page 503, the CCU located both the IMS segment and its corresponding DB2 row, but the non-key data of the listed field/column has a different value. The CCU considers this MISMATCHING DATA and it:

- Generates an SQL UPDATE statement for that row.
- Generates a DL/I REPL statement for that segment.
- Writes these statements to the repair data sets.
- Prints message EKYC935I.

```
EKYC935I MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."T5100">

         IMS KEY/ID FIELDS TO SEGMENT S5100
          KEY FIELD K0000   IN SEGMENT S0000   +0000   F2F2F1F6 F0F5       *221605        *
          KEY FIELD K5000   IN SEGMENT S5000   +0000   F0F5               *05            *
          ID  FIELD F5000   IN SEGMENT S5000   +0000   F0F5               *05            *
          ID  FIELD F5001   IN SEGMENT S5000   +0000   F0F540             *05            *
          ID  FIELD F5002   IN SEGMENT S5000   +0000   00000000 0000500C   *......&.      *
          KEY FIELD K5100   IN SEGMENT S5100   +0000   F0F5               *05            *

         DB2 PRIMARY KEY
          KEY COLUMN <K0000>                   +0000   F2F2F1F6 F0F5       *221605        *
          KEY COLUMN <K5000>                   +0000   F0F5               *05            *
          KEY COLUMN <K5001>                   +0000   F0F5               *05            *
          KEY COLUMN <K5002>                   +0000   F0F540             *05            *
          KEY COLUMN <K5003>                   +0000   00000000 0000500C   *......&.      *
          KEY COLUMN <K5100>                   +0000   F0F5               *05            *

         HAVE MISMATCHING NON-KEY DATA

          IMS FIELD F5100    IN SEGMENT S5100  +0000   C1C2C3C4            *ABCD          *
          DB2 COLUMN C5100                     +0000   C1C2C2C2            *ABBB          *

         ** GENERATED REPAIR STATEMENT NUMBER:             3 **
```

Figure 379. Mismatching Data

Message EKYC935I also provides the following information:

- The name of the DB2 table or view for which the error was detected.
- The segment name and the IMS segment key in a 16-byte dump format. The printed key consists of:
  - The value of the DB PCB key feedback area to the segment.
  - Eventual additional SSA fields (ID fields) declared to IMS DPROP to make the IMS key unique.
- The DB2 row key in a 16-byte dump format. The key is the DB2 primary key of the table.
- A line indicating that a non-key mismatch was encountered.
- A pair of message lines in 16-byte dump format describing the fields and columns in error.
- A line identifying the generated repair statement number. This number corresponds to the repair statement in the //CCUDERR or //CCUIERR repair files, or both.

EKYC935I and EKYC936I

If you are checking a mapping case 3 propagation request involving an internal segment and receive message EKYC935I, you might also receive message EKYC936I, as illustrated in Figure 380 on page 504. EKYC936I tells you that the CCU will not generate a DL/I repair statement.

If the CCU cannot find a mapping case 3 internal segment, it uses the containing segment to either insert, delete, or update the internal segment. However, if the containing segment is not available, the CCU cannot generate the DL/I repair statement.

However, the CCU did generate the DB2 repair statement successfully.

```
EKYC935I MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."T5100B">

         IMS KEY/ID FIELDS TO SEGMENT S5100
          KEY FIELD K0000    IN SEGMENT S0000    +0000   F2F2F1F6 F0F5       *221605       *
          KEY FIELD K5000    IN SEGMENT S5000    +0000   F0F5               *05           *
          ID  FIELD F5000    IN SEGMENT S5000    +0000   F0F5               *05           *
          ID  FIELD F5001    IN SEGMENT S5000    +0000   F0F540             *05           *
          ID  FIELD F5002    IN SEGMENT S5000    +0000   00000000 0000500C  *......&.      *
          KEY FIELD K5100    IN SEGMENT S5100    MISSING
              FIELD F5101    IN SEGMENT S5100B   MISSING

         DB2 PRIMARY KEY
          KEY COLUMN <K0000>                     +0000   F2F2F1F6 F0F5       *221605       *
          KEY COLUMN <K5000>                     +0000   F0F5               *05           *
          KEY COLUMN <K5001>                     +0000   F0F5               *05           *
          KEY COLUMN <K5002>                     +0000   F0F540             *05           *
          KEY COLUMN <K5003>                     +0000   00000000 0000500C  *......&.      *
          KEY COLUMN <K5100>                     +0000   F0F5               *05           *
          KEY COLUMN <K5101>                     +0000   0001               *..           *

         WITHOUT CORRESPONDING OCCURRENCE OF INTERNAL IMS SEGMENT S5100B

         ** GENERATED REPAIR STATEMENT NUMBER:             1 **

EKYC936I DL/I REPAIR STATEMENT SUPPRESSED FOR MAPPING CASE 3 INTERNAL SEGMENT OF PR PR5100B
         BECAUSE ITS (CONTAINING) IMS SEGMENT COULD NOT BE LOCATED IN THE IMS DATABASE
```

Figure 380. Message EKYC935I and EKYC936I

If you receive message EKYC936I, determine why the containing segment (S5100 in Figure 380) is missing. You can submit the CCU again to check the propagation request that describes the containing segment as an entity segment.

EKYC956W

The CCU issues message EKYC956W, shown in Figure 381, if it tries to create a repair statement for a DB2 row that contains a floating point column, and the data could not be converted to floating point.

```
EKYC956W FLOATING POINT CONVERSION ERROR ON COLUMN NAME <"CFLOAT">,
         ZERO VALUE DEFINED ON REPAIR STATEMENT

EKYC935I MISMATCH BETWEEN ...
         IMS KEY/ID FIELDS TO SEGMENT
         ...
         DB2 PRIMARY KEY
         ...
```

Figure 381. Messages EKYC956W and EKYC935I

The CCU continues processing, but you must verify the generated SQL statement in the DB2 repair data set. The column that should contain a floating point value is set to zero, regardless of what the real value is.

The Generated Repair Data Sets

The CCU generates a DL/I repair call (if the propagation request is hierarchical to relational or two way), and an SQL repair statement (if the propagation request is relational to hierarchical or two way). The CCU stores the repair calls in the repair data sets if:

- You ran either the error location phase, or the read phase of the direct technique, and specified the READONLY keyword on the CHECK control statement.
- The CCU identifies data mismatches for the data to be checked.
- A data mismatch report is printed.

If you are using the hashing technique to read the IMS database and DB2 tables, and if the IMS segment is propagated through a limited or full function propagation request (PRTYPE=L or F), then the error location program might not be able to locate a missing IMS segment. This is because the key of the IMS segment cannot be constructed from the DB2 row key (backward mapping is not possible). Such an occurrence is reported as a PROBABLE DATA MISMATCH on //CCUPRINT.

As indicated in 494, not all data inconsistencies are actual errors. The CCU does not create repair statements in these instances.

The CCU does not consider a missing IMS extension segment to be a mismatch if the columns representing this extension segment in the target DB2 table have a NOT NULL WITH DEFAULT definition.

Figure 382 shows a sample data mismatch report.

```
EKYC935I MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."T5100">

         IMS KEY/ID FIELDS TO SEGMENT S5100
          KEY FIELD K0000    IN SEGMENT S0000       +0000   F2F2F1F6 F0F5                        *221605          *
          KEY FIELD K5000    IN SEGMENT S5000       +0000   F0F5                                 *05              *
          ID  FIELD F5000    IN SEGMENT S5000       +0000   F0F5                                 *05              *
          ID  FIELD F5001    IN SEGMENT S5000       +0000   F0F540                               *05              *
          ID  FIELD F5002    IN SEGMENT S5000       +0000   00000000 0000500C                    *......&.        *
          KEY FIELD K5100    IN SEGMENT S5100       +0000   F0F5                                 *05              *

         WITHOUT CORRESPONDING ROW IN TABLE

         ** GENERATED REPAIR STATEMENT NUMBER:              1 **
-------------------------------------------------------------------------------------------------------------------
EKYC935I MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."T5100">

         DB2 PRIMARY KEY
          KEY COLUMN <C0000>                        +0000   F2F2F1F6 F0F6                        *221606          *
          KEY COLUMN <C5000>                        +0000   F0F6                                 *06              *
          KEY COLUMN <C5001>                        +0000   F0F6                                 *06              *
          KEY COLUMN <C5002>                        +0000   F0F640                               *06              *
          KEY COLUMN <C5003>                        +0000   00000000 0000600C                    *......&.        *
          KEY COLUMN <C5100>                        +0000   F0F6                                 *06              *

         WITHOUT CORRESPONDING IMS SEGMENT S5100 IN THE DATABASE

         ** GENERATED REPAIR STATEMENT NUMBER:              2 **
-------------------------------------------------------------------------------------------------------------------
EKYC935I MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."T5100">

         IMS KEY/ID FIELDS TO SEGMENT S5100
          KEY FIELD K0000    IN SEGMENT S0000       +0000   F2F2F1F6 F0F7                        *221607          *
          KEY FIELD K5000    IN SEGMENT S5000       +0000   F0F7                                 *07              *
          ID  FIELD F5000    IN SEGMENT S5000       +0000   F0F7                                 *07              *
          ID  FIELD F5001    IN SEGMENT S5000       +0000   F0F740                               *07              *
          ID  FIELD F5002    IN SEGMENT S5000       +0000   00000000 0000700C                    *......&.        *
          KEY FIELD K5100    IN SEGMENT S5100       +0000   F0F7                                 *07              *

         DB2 PRIMARY KEY
          KEY COLUMN <C0000>                        +0000   F2F2F1F6 F0F7                        *221607          *
          KEY COLUMN <C5000>                        +0000   F0F7                                 *07              *
          KEY COLUMN <C5001>                        +0000   F0F7                                 *07              *
          KEY COLUMN <C5002>                        +0000   F0F740                               *07              *
          KEY COLUMN <C5003>                        +0000   00000000 0000700C                    *......&.        *
          KEY COLUMN <C5100>                        +0000   F0F7                                 *07              *

         HAVE MISMATCHING NON-KEY DATA

          IMS FIELD F5100VC  IN SEGMENT S5100       +0000   C1C2C3C4                             *ABCD            *
          DB2 COLUMN C5100VC                        +0000   C1C2C2C2                             *ABBB            *

         ** GENERATED REPAIR STATEMENT NUMBER:              3 **
-------------------------------------------------------------------------------------------------------------------
```

Figure 382. Generated Data Mismatch Report

Repair Data Set for the DB2 Tables

In Figure 382 on page 505 the CCU detected three mismatches, as indicated by three EKYC935I messages.

In the first EKYC935I message, an IMS segment was retrieved from the database, but the corresponding DB2 row was missing. Figure 383 shows the SQL INSERT statement that was generated for this error.

```
*INSERT INTO DB2PROD."T5100"          (
*"C0000"             ,
*"C5000"             ,
*"C5001"             ,
*"C5002"             ,
*"C5003"             ,
*"C5100"             ,
*"C5100VC"           )
*VALUES(
*'221605'
*'05',
*'05',
*'05 ',
*5.0,
*'05',
*'missing data'
*);
```

Figure 383. SQL INSERT Statement

In the second message, a DB2 row was retrieved from the table, but the corresponding IMS segment is not in the database. The key of the DB2 table DB2PROD.″T5100″, is listed, and the missing segment is one of the S5100 segment types. Figure 384 shows the SQL DELETE statement generated for this error.

```
*** GENERATED REPAIR STATEMENT NUMBER:            2 **
*DELETE FROM DB2PROD."T5100"
*WHERE
*"C0000"                                =
*'221606'
*AND
*"C5000"                                =
*'06'
*AND
*"C5001"                                =
*'06'
*AND
*"C5002"                                =
*'06 '
*AND
*"C5003"                                =
*6.0
*AND
*"C5100"                                =
*'06'
*;
```

Figure 384. SQL DELETE Statement

In the third message, the CCU found both the segment and row, but the propagated non-key data has different values. The message lists both the keys of the segment

and row, and the field and column that have mismatching data. Figure 385 shows the SQL UPDATE statement generated for this mismatch.

```
*** GENERATED REPAIR STATEMENT NUMBER:            3 **
*UPDATE      DB2PROD."T5100"
*SET
*"C5100VC"                                =
*'ABCD         '
*WHERE
*"C0000"                                  =
*'221607'
*AND
*"C5000"                                  =
*'07'
*AND
*"C5001"                                  =
*'07'
*AND
*"C5002"                                  =
*'07 '
*AND
*"C5003"                                  =
*7.0
*AND
*"C5100"                                  =
*'07'
*;
```

Figure 385. SQL UPDATE Statement

The generated SQL statements contain the following information:

- The first line includes the access type (INSERT, DELETE, or UPDATE) and the DB2 table name. Normally, the table name consists of a table qualifier and a table name.

  If eight asterisks (*) appear instead of a table qualifier, the CCU did not receive a valid table qualifier from the IMS DPROP mapping tables. The CCU generated asterisks to prevent you from unintentionally updating an incorrect DB2 table. You must overwrite the asterisks with a valid DB2 table qualifier.
- Every SQL statement, ends with a semicolon (;).
- For an INSERT statement, there is a list of the columns to be inserted. Values for every listed column follow the list.
- For a DELETE statement, there is a list of the names of the key columns, and their values.
- For an UPDATE statement there is a list of the names of the data columns that must be updated and their values.
- In the WHERE clause, there is a list of the names of the key columns and their values for the row that must be updated.

Editing The DB2 Repair Data Set

After the CCU generates the DB2 repair data set, you can edit and sort it according to your installation's requirements.

The first line in the DB2 repair data set indicates the date and time the repair data set was generated, for example:

```
**** CCU REPAIR FILE - 1993-01-01-12.00.00.000000 ***
```

Each line that is the subject of an SQL statement has an asterisk (*) in the first position.

Note that when the DB2 System parameter IMS DPROP SUPPORT is set to 2, you must make all updates to the DB2 tables in an IMS environment, including those made by the repair data set. You can do so with either a user-written program or the IMS DPROP-provided EKYTIAD program, whichacts as an IMS application. EKYTIAD calls DSNTIAD, so the CCU can process the SQL statements in the repair data set.

If you don't change the repair data set before you pass it to the DSNTEP2 or DSNTIAD program, then the entire data set is read as comments only or invalid SQL statements. Refer to DB2 Administration Guide for more information on DSNTEP2 and DSNTIAD.

The asterisks force you to verify all the generated SQL repair statements with your data in the IMS database and the DB2 tables for which the SQL statements are generated. To submit this data set to the DSNTEP2 or DSNTIAD program, the SQL statements must be between columns 1 and 72 in each line. The SQL statements generated by the CCU start in column 2 and end in column 73. To have executable SQL statements, you must *shift* all lines of the SQL statement you want to execute one byte to the left to activate them. Use the ISPF/PDF EDIT to shift the lines. Before you shift the lines in the SQL statements, you must sort them. The CCU creates the SQL repair data statements in no particular order when there is a data mismatch. The CCU generates and writes the repair statements regardless of whether there are dependencies, and it does not insert a child row if there is no parent row. Therefore, you should sort the repair statements so the CCU inserts all parent statements before any child statements. For example:

In the CCU program buffers, there are two rows, one for table T1 and one for table T2. RIRs exist for the tables, T2 is the "child" table of T1. The CCU detects that a row in table T2 is missing, generates an INSERT INTO T2 statement, and writes it to the repair data set. Later, the CCU detects that the "parent" row for the missing T2 child row is also missing, generates an INSERT INTO T1 statement, and writes it to the repair data set.

To sort the repair data set, include the following sort step in the JCL:

```
SORT FIELDS=(74,7,CH,A)
```

You sort on a 7 byte long sort field that the CCU generates. The sort field is in every line of the repair data set between columns 74 and 80, before you make any changes to the repair data set.

Repair Data Set for the IMS Database

Figure 382 on page 505 shows that the CCU generated three repair statements for data mismatches. There is an EKYC935I message for each mismatch.

In the first EKYC935I message, the CCU retrieved an IMS segment from the database, but the corresponding DB2 row was missing. In the DB2 repair data set, an SQL INSERT statement is created for the missing DB2 row (see Figure 383 on page 506). In the IMS repair data set, the CCU generated a DL/I DLET call as shown in Figure 386 on page 509.

```
.** GENERATED REPAIR STATEMENT NUMBER:      1 **
U        GU    S0000   (K0000   = 221605)
EH8    OK
U        GU    S5000   (K5000   = 05&000   = 05&F5001   = 05 &F5002  *
N               = ......&.)
U        GHU   S5100   (K5100   = 05)
U        DLET
E 8
```

Figure 386. DL/I DLET Call

For this data mismatch, you must decide whether to use the DB2 insert call or the DL/I delete call to fix the inconsistency between the IMS and DB2 data. If you use both the DB2 and the DL/I repair calls, then you transfer the data mismatch from the DB2 table to the IMS database.

In the second message in Figure 382 on page 505, the CCU retrieved a DB2 row from the table, but the corresponding segment is not in the IMS database. The message lists the key of the DB2 table DB2PROD.″T5100″. The missing segment is one of the S5100 segment types. Figure 387 shows the DL/I ISRT call generated for this mismatch.

```
.** GENERATED REPAIR STATEMENT NUMBER:      2 **
U        GU    S0000   (K0000   = 221606)
EH8    OK
U        GU    S5000   (K5000   = 06&F5000   = 06&F5001   = 06 &F5002  *
N               = ......&.)
U        ISRT  S5100
U        DATA  06...converted non-key data from the DB2 row...
E 8
```

Figure 387. DL/I ISRT Call

The IMS repair data set contains an ISRT call. Figure 384 on page 506 shows an SQL statement to delete the corresponding DB2 row from table DB2PROD.″T5100″. You must decide whether to use the DB2 delete call or the IMS insert call to fix the inconsistency. Remember that inserting a logical child segment requires that its logical parent exists.

In the third message in Figure 382 on page 505, the CCU found both the segment and the row, but the propagated non-key data has different values. The message contains the keys of the segment and the row, and the field and its column that have mismatching data. Figure 388 shows the DL/I REPL call generated for this example.

```
.** GENERATED REPAIR STATEMENT NUMBER:      3 **
U        GU    S0000   (K0000   = 221607)
EH8    OK
U        GU    S5000   (K5000   = 07&F5000   = 07&F5001   = 07 &F5002  *
N               = ......&.)
U        GHU   S5100   (K5100   = 07)
U        REPL  S5100
U        DATA  07...converted non-key data from the DB2 row...
E 8
```

Figure 388. DL/I REPL Call

Either the IMS REPL call or the DB2 UPDATE call (see Figure 385 on page 507) can fix the non-key data mismatch. You must decide whether to use the repair call from the DB2 data set or the repair call from the IMS repair data set.

The structure of the generated DL/I call corresponds to the control statement description for the DL/I Test Program. For more information on the DL/I Test Program, DFSDDLT0, see*IMS/ESA Application Programming: DL/I Calls*.

Editing The IMS Repair Data Set

After the CCU generates the IMS repair data set, you can edit and sort it according to your installation's requirements.

The IMS repair data set consists of a status statement, comment statements, and compare statements.

- Status statement

  The status statement is indicated by S in first position. It is used to define the IMS database physical DBD name used to repair the data mismatches in the database and appears as follows:

  ```
  S11 2 2 2 2    DEDBRN07
  ```

- Comment statements

  Comment statements begin with a U, an N, or a period (.) in the first position. There are two types of comment statements: comments that are printed when you submit the DFSDDLT0 job step, and comments not printed.

  Unprinted comments structure every repair action. These comments are:

```
.**** CCU REPAIR FILE - 1993-01-01-12.00.00.000000 ***
.*
.** GENERATED REPAIR STATEMENT NUMBER:      1 **
```

Figure 389. Unprinted Comment

  The first comment line indicates the date and time the repair data set was generated. The second comment line describes a sequence number of the DL/I calls following the comment line. This sequence number corresponds to the data mismatch report lines printed on the //CCUPRINT data set.

  The second comment type, the printed comment, describes the suggested repair action to fix a data mismatch if you decide to use the DL/I call instead of the SQL statement to fix data inconsistencies.

  If you use the IMS repair data set to the DFSDDLT0 program without changing it, then DFSDDLT0 treats all DL/I calls as comments only. The U and N in first position force you to:

  – Verify all generated DL/I repair calls with your data in the IMS database and the DB2 table.
  – Compare the DL/I calls with the DB2 repair statements in the generated DB2 repair data set.

    To have executable DL/I calls, you must *replace* the U in the first position of a DL/I call line you want to execute with an L. Any N in first position, following a U, indicates a continuation line that must be *replaced* with a "blank." Use the ISPF/PDF editor to change the lines.

- Compare statements

For each DL/I call, the CCU generates a compare statement that allows the DFSDDLT0 program terminate when an unexpected DL/I status code is returned by IMS.

The repair file gets additional SYNC calls to release Fast Path buffers if it is a DEDB database.

If the repair call describes a segment propagated by a mapping case 3 propagation request, analyze the DL/I repair file very carefully. The CCU could have generated more than one repair call addressing the same occurrence of the IMS segment.

It is recommended that you sort the generated DL/I statements before you submit them to DFSDDLT0. See 508 for an explanation.

The CCU generates a 7-byte sort field in every line of the repair data set, between columns 74 and 80. To sort the repair data set, submit a job with a sort step as follows:

```
SORT FIELDS=(74,7,CH,A)
```

Sample CCU Reports for the Hashing Technique

This topic provides a sample hashing technique report for a full CCU run.

The report shown in Figure 390 on page 512 contains messages from CCU initialization, IMS read, DB2 read, hash sum compare, compare, and error location phases.

```
DPROP120   PGM=EKYC000X   JOB/STEP=JOB1     INIT                DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC000I INITIALIZATION PHASE STARTED

EKYC010I CHECK STATEMENT ENTERED BY THE USER FOLLOWS THIS MESSAGE:

         CHECK DBD=PRODDB,
                  SEG=(S0000,S1000,S1300,S3000,S5100),
                  ASSIGN=(S5100=PR5100A,S5100=PR5100B),
               MAXERROR=100,
               PSBNAME=EKYCCU00, PROCOPT=G
             ;
EKYC047I CCU USES DBNAME PRODDB PRSET NAME PRSET1, RECEIVED FROM THE DPROP DIRECTORY

EKYC059I TWO PRIDS, PR5100A AND PR5100B, REFER TO SEGMENT S5100.
         THE CCU WILL PROCESS BOTH PRIDS, BUT IN CASE OF DATA INCONSISTENCIES,
         THE REPAIR STATEMENTS FOR THESE PRIDS MIGHT BE CONFLICTING
EKYC072I DATABASE PRODDB CONTAINS THE FOLLOWING PRIDS, SEGMENTS AND PROPAGATED TABLES:

                  PR-  MAPPING-
         PRID     TYPE CASE DIR SEGMENT  ROLE PARENT   DB2 TABLE NAME              CHECK COMMENTS
         -------- ---- ---- --- -------- ---- -------- --------------------------- ----- -------------------------------
         PR0000   E    1    TW  S0000    E             DB2PROD.T0000               YES   IN SEG= LIST
         PR1000   E    2    TW  S1000    E    S0000    DB2PROD.T1000               YES   IN SEG= LIST
         PR1000   E    2    TW  S1300    X    S1000    DB2PROD.T1000               YES   IN SEG= LIST
         PR1100   E    1    TW  S1100    E    S1000    DB2PROD.T1100               NO
         PR2000   E    1    TW  S2000    E    S0000    DB2PROD.T2000               NO
         PR3000   E    1    TW  S3000    E    S0000    DB2PROD.T3000               YES   IN SEG= LIST
         PR4000   E    1    TW  S4000    E    S0000    DB2PROD.T4000               NO
         PR5000   E    1    TW  S5000    E    S0000    DB2PROD.T5000               NO
         PR5100   E    2    TW  S5100    E    S5000    DB2PROD.T5100               NO    IN ASSIGN= LIST, OTHER PRID
         PR5100   E    2    TW  S5101    X    S5100    DB2PROD.T5100               NO
         PR5100A  E    3    TW  S5100    C    S5000    DB2PROD.T5100A              YES   IN SEG= AND ASSIGN= LIST
         PR5100A  E    3    TW  S5100A   E    S5100    DB2PROD.T5100A              YES   INTERNAL SEGMENT
         PR5100B  E    3    TW  S5100    C    S5000    DB2PROD.T5100B              YES   IN SEG= AND ASSIGN= LIST
         PR5100B  E    3    TW  S5100B   E    S5100    DB2PROD.T5100B              YES   INTERNAL SEGMENT
         PR5100C  E    3    TW  S5100    C    S5000    DB2PROD.T5100C              NO    IN ASSIGN= LIST, OTHER PRID
         PR5100C  E    3    TW  S5100C   E    S5100    DB2PROD.T5100C              NO    INTERNAL SEGMENT
EKYC085I THE FOLLOWING PSB SOURCE CODE IS ASSEMBLED AND LINKED INTO IMSVS.PSBLIB:

         CCUPCB   PCB   TYPE=DB,SB=COND,DBDNAME=PRODDB,PROCOPT=G,               C
                        PCBNAME=CCUPCB,KEYLEN=00017
                  SENSEG NAME=S0000,PARENT=0
                  SENSEG NAME=S1000,PARENT=S0000
                  SENSEG NAME=S1300,PARENT=S1000
                  SENSEG NAME=S3000,PARENT=S0000
                  SENSEG NAME=S5000,PARENT=S0000
                  SENSEG NAME=S5100,PARENT=S5000
                  PSBGEN LANG=ASSEM,PSBNAME=EKYCCU00,                           C
                        IOEROPN=(451,WTOR),CMPAT=YES
                  END
EKYC095I SQL STATEMENT FOR PRID PR0000, TABLE <DB2PROD."T0000">:
SELECT "K0000", "C0000" FROM DB2PROD."T0000"

EKYC095I SQL STATEMENT FOR PRID PR1000, TABLE <DB2PROD."T1000">:
         SELECT "K0000", "K1000", "C1000", "C1300" FROM
            DB2PROD."T1000"

EKYC095I SQL STATEMENT FOR PRID PR3000, TABLE <DB2PROD."T3000">:
         SELECT "K0000", "K3000", "C3000", "C3000IN1", "C3000IN2",
            "C3000DC1", "C3000DC2", "C3000TMS" FROM DB2PROD."T3000"

EKYC095I SQL STATEMENT FOR PRID PR5100A, TABLE <DB2PROD."T5100A">:
         SELECT "K0000", "K5000", "K5001", "K5002", "K5003", "K5100",
            "C5100VG" FROM DB2PROD."T5100A"

EKYC095I SQL STATEMENT FOR PRID PR5100B, TABLE <DB2PROD."T5100B">:
         SELECT "K0000", "K5000", "K5001", "K5002", "K5003", "K5100",
            "C5100", "C5100TIM", "K5101" FROM DB2PROD."T5100B"
```

Figure 390. Sample CCU Report: Hashing Technique (Part 1 of 4)

```
EKYC002I CONTROL DATA SET SUCCESSFULLY CREATED.
TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING HASHING TECHNIQUE.
         DPROP SYSTEM NAME=DPROP001 TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS

EKYC003I INITIALIZATION PHASE ENDED NORMALLY.
==================================================================================================================

DPROP120   PGM=EKYC100X   JOB/STEP=JOB1     G        IMSREAD    DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC100I DL/I READ PHASE STARTED

EKYC815I CONTROL DATA SET SPECIFICATIONS:
         TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING HASHING TECHNIQUE.
         DPROP SYSTEM NAME=DPROP001 TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS
EKYC828I THE DPROP DIRECTORY INDICATES ERROPT(PROPAGATION FAILURE)=IGNORE
OR A STATUS OTHER THAN 'ACTIVE' FOR THE FOLLOWING PRIDS:

         PRID     STATUS    ERROPT  SEGMENT  DB2 TABLE NAME
         -------- --------- ------- -------- ---------------------------------------------------
         PR0000   ACTIVE    IGNORE  S0000    DB2PROD."T0000"
         PR5100A  INACTIVE  BACKOUT S5100A   DB2PROD."T5100A"
         PR5100B  SUSPENDED IGNORE  S5100B   DB2PROD."T5100B"
EKYC868I STATISTICS FOR RETRIEVED DL/I SEGMENTS OF DATABASE PRODDB:

                                                   SEGMENT TYPE IS PROPAGATED BY              NUMBER OF
         TYPE          RETRIEVED     CHECK     PRID     MAPPING CASE/OPTIONS    WITH ROLE   RESULTING ROWS
         -----------------------     -----     ----------------------------------------- --------------
         S0000                 4     YES       PR0000   MC1                     ENTITY                  4
                                               PR5100A  MC3 / PATHDATA          PARENT                  -
                                               PR5100B  MC3 / PATHDATA          PARENT                  -
         S1000                 4     YES       PR1000   MC2                     ENTITY                  4
         S1300                 2     YES       PR1000   MC2                     EXTENSION               4
         S3000                 9     YES       PR3000   MC1                     ENTITY                  9
         S5000                 4               PR5100A  MC3 / PATHDATA          PARENT                  -
                                               PR5100B  MC3 / PATHDATA          PARENT                  -
         S5100                 6     YES       PR5100A  MC3 / PATHDATA          CONTAINING              -
           S5100A                    YES       PR5100A  MC3                     ENTITY                 10
                                     YES       PR5100B  MC3 / PATHDATA          CONTAINING              -
           S5100B                    YES       PR5100B  MC3                     ENTITY                  5

EKYC102I DL/I READ PHASE ENDED AT END OF DATA.
==================================================================================================================

DPROP120   PGM=EKYC200X   JOB/STEP=JOB1     DB2READ             DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC200I DB2 READ PHASE STARTED

EKYC815I CONTROL DATA SET SPECIFICATIONS:
         TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING HASHING TECHNIQUE.
         DPROP SYSTEM NAME=DPROP001 TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS
EKYC828I THE DPROP DIRECTORY INDICATES ERROPT(PROPAGATION FAILURE)=IGNORE
OR A STATUS OTHER THAN 'ACTIVE' FOR THE FOLLOWING PRIDS:

         PRID     STATUS    ERROPT  SEGMENT  DB2 TABLE NAME
         -------- --------- ------- -------- ---------------------------------------------------
         PR0000   ACTIVE    IGNORE  S0000    DB2PROD."T0000"
         PR5100A  INACTIVE  BACKOUT S5100A   DB2PROD."T5100A"
         PR5100B  SUSPENDED IGNORE  S5100B   DB2PROD."T5100B"
```

Figure 390. Sample CCU Report: Hashing Technique (Part 2 of 4)

```
EKYC875I STATISTICS FOR RETRIEVED ROWS FROM DB2 TABLES:

          ROWS RETRIEVED DB2 TABLE NAME                                    PRID     MAPPING CASE / OPTIONS
         --------------- ------------------------------------------------- -------- ----------------------
                       4 DB2PROD."T0000"                                   PR0000   MC1
                       4 DB2PROD."T1000"                                   PR1000   MC2
                       9 DB2PROD."T3000"                                   PR3000   MC1
                      10 DB2PROD."T5100A "                                 PR5100A  MC3 / PATHDATA
                       6 DB2PROD."T5100B "                                 PR5100B  MC3 / PATHDATA

EKYC202I DB2 READ PHASE ENDED AT END OF DATA
==================================================================================================================

DPROP120   PGM=EKYC400X   JOB/STEP=JOB1     HSUMCOMP            DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1

EKYC400I HASH SUM COMPARE PHASE STARTED

EKYC432I HASH SUMS ARE     EQUAL FOR PRID PR0000   SEGMENT S0000    TABLE <DB2PROD."T0000">

EKYC432I HASH SUMS ARE     EQUAL FOR PRID PR1000   SEGMENT S1000    TABLE <DB2PROD."T1000">

EKYC432I HASH SUMS ARE     EQUAL FOR PRID PR3000   SEGMENT S3000    TABLE <DB2PROD."T3000">

EKYC432I HASH SUMS ARE     EQUAL FOR PRID PR5100A  SEGMENT S5100A   TABLE <DB2PROD."T5100A">

EKYC431W HASH SUMS ARE NOT EQUAL FOR PRID PR5100B  SEGMENT S5100B   TABLE <DB2PROD."T5100B">

EKYC496I HASH SUMS OF DL/I AND DB2 READ PHASES ARE NOT EQUAL

EKYC498I HASH SUM COMPARE PHASE ENDED WITH WARNINGS
==================================================================================================================

DPROP120   PGM=EKYC500X   JOB/STEP=JOB1     COMPARE             DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1

EKYC500I COMPARE PHASE STARTED

EKYC580I 12 RECORDS FROM //CCUHASH1 AND //CCUHASH2 DATA SETS READ

EKYC581I 10 RECORDS FROM //CCUHASH1 AND //CCUHASH2 DATA SETS DELETED

EKYC582I 2 MISMATCH INDICATION RECORDS WRITTEN TO THE //CCUMSMTC DATA SET

EKYC597I COMPARE PHASE ENDED NORMALLY
==================================================================================================================

DPROP120   PGM=EKYC600X   JOB/STEP=JOB1     G        ERRORLOC   DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC600I ERROR LOCATION PHASE STARTED

EKYC815I CONTROL DATA SET SPECIFICATIONS:
         TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING HASHING TECHNIQUE.
         DPROP SYSTEM NAME=DPROP001 TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS
EKYC828I THE DPROP DIRECTORY INDICATES ERROPT(PROPAGATION FAILURE)=IGNORE
OR A STATUS OTHER THAN 'ACTIVE' FOR THE FOLLOWING PRIDS:

         PRID     STATUS    ERROPT  SEGMENT  DB2 TABLE NAME
         -------- --------- ------- -------- ------------------------------------------------
         PR0000   ACTIVE    IGNORE  S0000    DB2PROD."T0000"
         PR5100A  INACTIVE  BACKOUT S5100A   DB2PROD."T5100A"
         PR5100B  SUSPENDED IGNORE  S5100B   DB2PROD."T5100B"
```

Figure 390. Sample CCU Report: Hashing Technique (Part 3 of 4)

```
EKYC816I THE FOLLOWING IMS FIELD AND DB2 COLUMN DEFINITIONS MIGHT CAUSE DATA COMPARE PROBLEMS IN THE CCU.
FOR MORE INFORMATION, SEE 'NOTES' FOLLOWING THIS LIST.

         PRID     SEGMENT  FIELD    DATATYPE DB2 COLUMN NAME    DATATYPE NULLS  DB2 TABLE NAME                      NOTE
         -------- -------- -------- -------- ------------------ -------- ------ ----------------------------------- ----
         PR3000   S3000    F3000IN1 B        C3000IN1           SMALLINT YES    DB2PROD."T3000"                        2
                           F3000IN2 B        C3000IN2           INTEGER  YES                                           2
                           F3000IN3 B        C3000DC1           DECIMAL  YES                                           2
                           F3000PK  PACKED   C3000DC2           DECIMAL  W.DEF.                                        5
                           F3000TMS TIMESTMP C3000TMS           TIMESTMP W.DEF.                                        3
         PR5100A  S5100A   F5100LVG LONGVARG C5100VG            VARG     W.DEF. DB2PROD."T5100A"                       1
         PR5100B  S5100B   F5100TIM TIME     C5100TIM           TIME     W.DEF. DB2PROD."T5100B"                       4

         NOTES:  1) TRAILING BLANK CHARACTERS (X'40') OF THE FIELD AND COLUMN DATA WILL BE TRUNCATED BEFORE COMPARISON.
                 2) BLANK OR ZERO FIELD DATA COMPARED WITH DB2 COLUMN 'NULLS' WILL NOT BE CONSIDERED AS DATA MISMATCH.
                 3) TIMESTAMPS HAVE DIFFERENT LENGTH; ONLY THE COMMON TIMESTAMP DATA VALUE WILL BE USED FOR COMPARISON.
                 4) FIELD/COLUMN WILL NOT BE CHECKED BECAUSE THE DB2 COLUMN DEFINITION IS 'NOT NULL WITH DEFAULT'.
                 5) SCALE FACTORS ARE DIFFERENT; ONLY THE COMMON PART OF THE SCALE VALUE WILL BE USED FOR COMPARISON.
EKYC935I MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."T5100B">

         IMS KEY/ID FIELDS TO SEGMENT S5100
          KEY FIELD K0000    IN SEGMENT S0000       +0000   F2F2F1F6 F0F5                        *221605          *
          KEY FIELD K5000    IN SEGMENT S5000       +0000   F0F5                                 *05              *
          ID  FIELD F5000    IN SEGMENT S5000       +0000   F0F5                                 *05              *
          ID  FIELD F5001    IN SEGMENT S5000       +0000   F0F540                               *05              *
          ID  FIELD F5002    IN SEGMENT S5000       +0000   00000000 0000500C                    *......&.        *
          KEY FIELD K5100    IN SEGMENT S5100       +0000   F0F5                                 *05              *
              FIELD F5101    IN SEGMENT S5100B      MISSING

         DB2 PRIMARY KEY
          KEY COLUMN <K0000>                        +0000   F2F2F1F6 F0F5                        *221605          *
          KEY COLUMN <K5000>                        +0000   F0F5                                 *05              *
          KEY COLUMN <K5001>                        +0000   F0F5                                 *05              *
          KEY COLUMN <K5002>                        +0000   F0F540                               *05              *
          KEY COLUMN <K5003>                        +0000   00000000 0000500C                    *......&.        *
          KEY COLUMN <K5100>                        +0000   F0F5                                 *05              *
          KEY COLUMN <K5101>                        +0000   0001                                 *..              *

         WITHOUT CORRESPONDING OCCURRENCE OF INTERNAL IMS SEGMENT S5100B

         ** GENERATED REPAIR STATEMENT NUMBER:             1 **

-------------------------------------------------------------------------------------------------------------------------------
EKYC614I DATA MISMATCH STATISTICS:

         PRID     SEGMENT  DB2 TABLE NAME                                    REPORTED MISMATCHES
         -------- -------- ------------------------------------------------- -------------------
         PR0000   S0000    DB2PROD."T0000"                                                     0
         PR1000   S1000    DB2PROD."T1000"                                                     0
         PR3000   S3000    DB2PROD."T3000"                                                     0
         PR5100A  S5100A   DB2PROD."T5100A"                                                    0
         PR5100B  S5100B   DB2PROD."T5100B"                                                    1

EKYC617I 2 MISMATCH INDICATION RECORDS READ FROM //CCUMSMTC DATA SET

EKYC619I 1 DATA MISMATCH INDICATION RECORDS SKIPPED

EKYC615I 1 DATA MISMATCHES BETWEEN DL/I AND DB2 DATA LISTED WITHIN THIS RUN

EKYC602I ERROR LOCATION PHASE ENDED AT END OF DATA.
```

Figure 390. Sample CCU Report: Hashing Technique (Part 4 of 4)

Sample CCU Reports for the Direct Technique

This topic provides sample reports of a full CCU run with an IMS database and the related propagated DB2 tables using the direct technique.

The report shown in Figure 391 on page 516 contains messages from CCU initialization, IMS and DB2 read and compare, and the error location phase.

```
DPROP120   PGM=EKYC000X   JOB/STEP=JOB1     INIT                DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1
EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001
EKYC000I INITIALIZATION PHASE STARTED
EKYC010I CHECK STATEMENT ENTERED BY THE USER FOLLOWS THIS MESSAGE:

          CHECK DBD=PRODDB,
                   SEG=(S0000,S1000,S1300,S3000,S5100),
                   ASSIGN=(S5100=PR5100A,S5100=PR5100B),
                 DIRECT,
                 MAXERROR=100,
                 PSBNAME=EKYCCU00, PROCOPT=G
               ;
EKYC047I CCU USES DBNAME PRODDB PRSET NAME PRSET1, RECEIVED FROM THE DPROP DIRECTORY
EKYC059I TWO PRIDS, PR5100A AND PR5100B, REFER TO SEGMENT S5100.
THE CCU WILL PROCESS BOTH PRIDS, BUT IN CASE OF DATA INCONSISTENCIES,
         THE REPAIR STATEMENTS FOR THESE PRIDS MIGHT BE CONFLICTING
EKYC072I DATABASE PRODDB CONTAINS THE FOLLOWING PRIDS, SEGMENTS AND PROPAGATED TABLES:

                  PR-  MAPPING-
         PRID     TYPE CASE DIR SEGMENT  ROLE PARENT   DB2 TABLE NAME              CHECK COMMENTS
         -------- ---- ---- --- -------- ---- -------- --------------------------- ----- -------------------------------
         PR0000   E    1    TW  S0000    E             DB2PROD.T0000               YES   IN SEG= LIST
         PR1000   E    2    TW  S1000    E    S0000    DB2PROD.T1000               YES   IN SEG= LIST
         PR1000   E    2    TW  S1300    X    S1000    DB2PROD.T1000               YES   IN SEG= LIST
         PR1100   E    1    TW  S1100    E    S1000    DB2PROD.T1100               NO
         PR2000   E    1    TW  S2000    E    S0000    DB2PROD.T2000               NO
         PR3000   E    1    TW  S3000    E    S0000    DB2PROD.T3000               YES   IN SEG= LIST
         PR4000   E    1    TW  S4000    E    S0000    DB2PROD.T4000               NO
         PR5000   E    1    TW  S5000    E    S0000    DB2PROD.T5000               NO
         PR5100   E    2    TW  S5100    E    S5000    DB2PROD.T5100               NO    IN ASSIGN= LIST, OTHER PRID
         PR5100   E    2    TW  S5101    X    S5100    DB2PROD.T5100               NO
         PR5100A  E    3    TW  S5100    C    S5000    DB2PROD.T5100A              YES   IN SEG= AND ASSIGN= LIST
         PR5100A  E    3    TW  S5100A   E    S5100    DB2PROD.T5100A              YES   INTERNAL SEGMENT
         PR5100B  E    3    TW  S5100    C    S5000    DB2PROD.T5100B              YES   IN SEG= AND ASSIGN= LIST
         PR5100B  E    3    TW  S5100B   E    S5100    DB2PROD.T5100B              YES   INTERNAL SEGMENT
         PR5100C  E    3    TW  S5100    C    S5000    DB2PROD.T5100C              NO    IN ASSIGN= LIST, OTHER PRID
         PR5100C  E    3    TW  S5100C   E    S5100    DB2PROD.T5100C              NO    INTERNAL SEGMENT
EKYC085I THE FOLLOWING PSB SOURCE CODE IS ASSEMBLED AND LINKED INTO IMSVS.PSBLIB:

         CCUPCB   PCB   TYPE=DB,SB=COND,DBDNAME=PRODDB,PROCOPT=G,              C
                        PCBNAME=CCUPCB,KEYLEN=00017
                  SENSEG NAME=S0000,PARENT=0
                  SENSEG NAME=S1000,PARENT=S0000
                  SENSEG NAME=S1300,PARENT=S1000
                  SENSEG NAME=S3000,PARENT=S0000
                  SENSEG NAME=S5000,PARENT=S0000
                  SENSEG NAME=S5100,PARENT=S5000
                  PSBGEN LANG=ASSEM,PSBNAME=EKYCCU00,                           C
                        IOEROPN=(451,WTOR),CMPAT=YES
                  END
EKYC095I SQL STATEMENT FOR PRID PR0000, TABLE <DB2PROD."T0000">:
SELECT "K0000", "C0000" FROM DB2PROD."T0000" ORDER BY "K0000"

EKYC095I SQL STATEMENT FOR PRID PR1000, TABLE <DB2PROD."T1000">:
         SELECT "K0000", "K1000", "C1000", "C1300" FROM
            DB2PROD."T1000" ORDER BY "K0000", "K1000"

EKYC095I SQL STATEMENT FOR PRID PR3000, TABLE <DB2PROD."T3000">:
         SELECT "K0000", "K3000", "C3000", "C3000IN1", "C3000IN2",
            "C3000DC1", "C3000DC2", "C3000TMS" FROM DB2PROD."T3000"
            ORDER BY "K0000", "K3000"

EKYC095I SQL STATEMENT FOR PRID PR5100A, TABLE <DB2PROD."T5100A">:
         SELECT "K0000", "K5000", "K5001", "K5002", "K5003", "K5100",
            "C5100VG" FROM DB2PROD."T5100A"  ORDER BY "K0000", "K5000",
            "K5001", "K5002", "K5003", "K5100"
```

Figure 391. Sample CCU Report: Direct Technique (Part 1 of 4)

```
EKYC095I SQL STATEMENT FOR PRID PR5100B, TABLE <DB2PROD."T5100B">:
         SELECT "K0000", "K5000", "K5001", "K5002", "K5003", "K5100",
            "C5100", "C5100TIM", "K5101" FROM DB2PROD."T5100B" ORDER BY
            "K0000", "K5000", "K5001", "K5002", "K5003", "K5100",
            "K5101"
EKYC002I CONTROL DATA SET SUCCESSFULLY CREATED.
TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING DIRECT TECHNIQUE.
         DPROP SYSTEM NAME=DPROP001 TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS

EKYC003I INITIALIZATION PHASE ENDED NORMALLY.
=================================================================================================================

DPROP120   PGM=EKYC300X   JOB/STEP=JOB1     G        READ       DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC300I READ AND COMPARE PHASE STARTED

EKYC815I CONTROL DATA SET SPECIFICATIONS:
         TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING DIRECT TECHNIQUE.
         DPROP SYSTEM NAME=DPROP001 TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS
EKYC828I THE DPROP DIRECTORY INDICATES ERROPT(PROPAGATION FAILURE)=IGNORE
OR A STATUS OTHER THAN 'ACTIVE' FOR THE FOLLOWING PRIDS:

         PRID     STATUS    ERROPT  SEGMENT  DB2 TABLE NAME
         -------- --------- ------- -------- ---------------------------------------------------
         PR0000   ACTIVE    IGNORE  S0000    DB2PROD."T0000"
         PR5100A  INACTIVE  BACKOUT S5100A   DB2PROD."T5100A"
         PR5100B  SUSPENDED IGNORE  S5100B   DB2PROD."T5100B"
EKYC310I STATISTICS FOR RETRIEVED IMS AND DB2 DATA:

         SEGMENT       RETRIEVED CHECK PRID     SEGMENT ROLE  CONVERTED SEGM                          CCUMSMTC RECORDS
                                       MAPPING CASE/OPTIONS   RETRIEVED ROWS DB2 TABLE NAME
         ----------------------- ----- ---------------------- -------------- -------------------------------------------
         S0000                 4 YES   PR0000   ENTITY                    4                                          0
                                       MC1                                4 DB2PROD."T0000"
                                 USED  PR5100A  PARENT
                                 USED  PR5100B  PARENT
                                 ----- ---------------------- -------------- -------------------------------------------
         S1000                 4 YES   PR1000   ENTITY                    4                                          0
                                       MC2                                4 DB2PROD."T1000"
                                 ----- ---------------------- -------------- -------------------------------------------
         S1300                 2 YES   PR1000   EXTENSION                 4                                          0
                                       MC2                                - DB2PROD."T1000"
                                 ----- ---------------------- -------------- -------------------------------------------
         S3000                 9 YES   PR3000   ENTITY                    9                                          0
                                       MC1                                9 DB2PROD."T3000"
                                 ----- ---------------------- -------------- -------------------------------------------
         S5000                 4 USED  PR5100A  PARENT
                                 USED  PR5100B  PARENT
                                 ----- ---------------------- -------------- -------------------------------------------
         S5100                 6 YES   PR5100A  CONTAINING               10                                          0
                                       MC3 / PATHDATA                    10 DB2PROD."T5100A"
                                 YES   PR5100B  CONTAINING                5                                          2
                                       MC3 / PATHDATA                     7 DB2PROD."T5100B"
                                 ----- ---------------------- -------------- -------------------------------------------
EKYC302I 2 MISMATCH INDICATION RECORDS WRITTEN TO //CCUMSMTC DATA SET

EKYC304I READ AND COMPARE PHASE ENDED AT END OF DATA.
=================================================================================================================

DPROP120   PGM=EKYC600X   JOB/STEP=JOB1     G        READ       DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1

EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001

EKYC600I ERROR LOCATION PHASE STARTED
```

Figure 391. Sample CCU Report: Direct Technique (Part 2 of 4)

```
EKYC815I CONTROL DATA SET SPECIFICATIONS:
         TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING DIRECT TECHNIQUE.
         DPROP SYSTEM NAME=DPROP001 TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS
EKYC828I THE DPROP DIRECTORY INDICATES ERROPT(PROPAGATION FAILURE)=IGNORE
OR A STATUS OTHER THAN 'ACTIVE' FOR THE FOLLOWING PRIDS:

         PRID     STATUS    ERROPT  SEGMENT  DB2 TABLE NAME
         -------- --------- ------- -------- --------------------------------------------------
         PR0000   ACTIVE    IGNORE  S0000    DB2PROD."T0000"
         PR5100A  INACTIVE  BACKOUT S5100A   DB2PROD."T5100A"
         PR5100B  SUSPENDED IGNORE  S5100B   DB2PROD."T5100B"
EKYC816I THE FOLLOWING IMS FIELD AND DB2 COLUMN DEFINITIONS MIGHT CAUSE DATA COMPARE PROBLEMS IN THE CCU.
FOR MORE INFORMATION, SEE 'NOTES' FOLLOWING THIS LIST.

         PRID     SEGMENT  FIELD    DATATYPE DB2 COLUMN NAME    DATATYPE NULLS  DB2 TABLE NAME                      NOTE
         -------- -------- -------- -------- ------------------ -------- ------ ---------------------------------- ----
         PR3000   S3000    F3000IN1 B        C3000IN1           SMALLINT YES    DB2PROD."T3000"                        2
                           F3000IN2 B        C3000IN2           INTEGER  YES                                           2
                           F3000IN3 B        C3000DC1           DECIMAL  YES                                           2
                           F3000PK  PACKED   C3000DC2           DECIMAL  W.DEF.                                        5
                           F3000TMS TIMESTMP C3000TMS           TIMESTMP W.DEF.                                        3
         PR5100A  S5100A   F5100LVG LONGVARG C5100VG            VARG     W.DEF. DB2PROD."T5100A"                       1
         PR5100B  S5100B   F5100TIM TIME     C5100TIM           TIME     W.DEF. DB2PROD."T5100B"                       4

         NOTES:  1) TRAILING BLANK CHARACTERS (X'40') OF THE FIELD AND COLUMN DATA WILL BE TRUNCATED BEFORE COMPARISON.
                 2) BLANK OR ZERO FIELD DATA COMPARED WITH DB2 COLUMN 'NULLS' WILL NOT BE CONSIDERED AS DATA MISMATCH.
                 3) TIMESTAMPS HAVE DIFFERENT LENGTH; ONLY THE COMMON TIMESTAMP DATA VALUE WILL BE USED FOR COMPARISON.
                 4) FIELD/COLUMN WILL NOT BE CHECKED BECAUSE THE DB2 COLUMN DEFINITION IS 'NOT NULL WITH DEFAULT'.
                 5) SCALE FACTORS ARE DIFFERENT; ONLY THE COMMON PART OF THE SCALE VALUE WILL BE USED FOR COMPARISON.
EKYC935I MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."T5100B">

         IMS KEY/ID FIELDS TO SEGMENT S5100
          KEY FIELD K0000    IN SEGMENT S0000       +0000   F2F2F1F6 F0F5                        *221605          *
          KEY FIELD K5000    IN SEGMENT S5000       +0000   F0F5                                 *05              *
          ID  FIELD F5000    IN SEGMENT S5000       +0000   F0F5                                 *05              *
          ID  FIELD F5001    IN SEGMENT S5000       +0000   F0F540                               *05              *
          ID  FIELD F5002    IN SEGMENT S5000       +0000   00000000 0000500C                    *......&.        *
          KEY FIELD K5100    IN SEGMENT S5100       +0000   F0F5                                 *05              *
              FIELD F5101    IN SEGMENT S5100B      MISSING

         DB2 PRIMARY KEY
          KEY COLUMN <K0000>                        +0000   F2F2F1F6 F0F5                        *221605          *
          KEY COLUMN <K5000>                        +0000   F0F5                                 *05              *
          KEY COLUMN <K5001>                        +0000   F0F5                                 *05              *
          KEY COLUMN <K5002>                        +0000   F0F540                               *05              *
          KEY COLUMN <K5003>                        +0000   00000000 0000500C                    *......&.        *
          KEY COLUMN <K5100>                        +0000   F0F5                                 *05              *
          KEY COLUMN <K5101>                        +0000   0001                                 *..              *

         WITHOUT CORRESPONDING OCCURRENCE OF INTERNAL IMS SEGMENT S5100B

         ** GENERATED REPAIR STATEMENT NUMBER:             1 **

-------------------------------------------------------------------------------------------------------------------------------


EKYC614I DATA MISMATCH STATISTICS:

         PRID     SEGMENT  DB2 TABLE NAME                                    REPORTED MISMATCHES
         -------- -------- ------------------------------------------------- -------------------
         PR0000   S0000    DB2PROD."T0000"                                                     0
         PR1000   S1000    DB2PROD."T1000"                                                     0
         PR3000   S3000    DB2PROD."T3000"                                                     0
         PR5100A  S5100A   DB2PROD."T5100A"                                                    0
         PR5100B  S5100B   DB2PROD."T5100B"                                                    1
```

Figure 391. Sample CCU Report: Direct Technique (Part 3 of 4)

```
EKYC617I 2 MISMATCH INDICATION RECORDS READ FROM //CCUMSMTC DATA SET

EKYC619I 1 DATA MISMATCH INDICATION RECORDS SKIPPED

EKYC615I 1 DATA MISMATCHES BETWEEN DL/I AND DB2 DATA LISTED WITHIN THIS RUN

EKYC602I ERROR LOCATION PHASE ENDED AT END OF DATA.
```

Figure 391. Sample CCU Report: Direct Technique (Part 4 of 4)

Figure 392 on page 520 is a sample report of a full CCU run with an IMS database and the related propagated DB2 tables. The direct technique was used, and READONLY was specified in the CHECK control statement.

The report contains messages from CCU initialization and the IMS and DB2 read and compare phase, including the repair data set generation in the read and compare phase.

```
DPROP120   PGM=EKYC000X   JOB/STEP=JOB1     INIT                DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1
EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001
EKYC000I INITIALIZATION PHASE STARTED
EKYC010I CHECK STATEMENT ENTERED BY THE USER FOLLOWS THIS MESSAGE:

          CHECK DBD=PRODDB,
                   SEG=(S0000,S1000,S1300,S3000,S5100),
                   ASSIGN=(S5100=PR5100A,S5100=PR5100B),
                 DIRECT,
                 READONLY,
                 MAXERROR=100,
                 PSBNAME=EKYCCU00, PROCOPT=G
               ;
EKYC047I CCU USES DBNAME PRODDB PRSET NAME PRSET1, RECEIVED FROM THE DPROP DIRECTORY
EKYC059I TWO PRIDS, PR5100A AND PR5100B, REFER TO SEGMENT S5100.
THE CCU WILL PROCESS BOTH PRIDS, BUT IN CASE OF DATA INCONSISTENCIES,
         THE REPAIR STATEMENTS FOR THESE PRIDS MIGHT BE CONFLICTING
EKYC072I DATABASE PRODDB CONTAINS THE FOLLOWING PRIDS, SEGMENTS AND PROPAGATED TABLES:

                  PR-  MAPPING-
         PRID     TYPE CASE DIR SEGMENT  ROLE PARENT   DB2 TABLE NAME              CHECK COMMENTS
         -------- ---- ---- --- -------- ---- -------- --------------------------- ----- -------------------------------
         PR0000   E    1    TW  S0000    E             DB2PROD.T0000               YES   IN SEG= LIST
         PR1000   E    2    TW  S1000    E    S0000    DB2PROD.T1000               YES   IN SEG= LIST
         PR1000   E    2    TW  S1300    X    S1000    DB2PROD.T1000               YES   IN SEG= LIST
         PR1100   E    1    TW  S1100    E    S1000    DB2PROD.T1100               NO
         PR2000   E    1    TW  S2000    E    S0000    DB2PROD.T2000               NO
         PR3000   E    1    TW  S3000    E    S0000    DB2PROD.T3000               YES   IN SEG= LIST
         PR4000   E    1    TW  S4000    E    S0000    DB2PROD.T4000               NO
         PR5000   E    1    TW  S5000    E    S0000    DB2PROD.T5000               NO
         PR5100   E    2    TW  S5100    E    S5000    DB2PROD.T5100               NO    IN ASSIGN= LIST, OTHER PRID
         PR5100   E    2    TW  S5101    X    S5100    DB2PROD.T5100               NO
         PR5100A  E    3    TW  S5100    C    S5000    DB2PROD.T5100A              YES   IN SEG= AND ASSIGN= LIST
         PR5100A  E    3    TW  S5100A   E    S5100    DB2PROD.T5100A              YES   INTERNAL SEGMENT
         PR5100B  E    3    TW  S5100    C    S5000    DB2PROD.T5100B              YES   IN SEG= AND ASSIGN= LIST
         PR5100B  E    3    TW  S5100B   E    S5100    DB2PROD.T5100B              YES   INTERNAL SEGMENT
         PR5100C  E    3    TW  S5100    C    S5000    DB2PROD.T5100C              NO    IN ASSIGN= LIST, OTHER PRID
         PR5100C  E    3    TW  S5100C   E    S5100    DB2PROD.T5100C              NO    INTERNAL SEGMENT
EKYC085I THE FOLLOWING PSB SOURCE CODE IS ASSEMBLED AND LINKED INTO IMSVS.PSBLIB:

         CCUPCB   PCB   TYPE=DB,SB=COND,DBDNAME=PRODDB,PROCOPT=G,              C
                        PCBNAME=CCUPCB,KEYLEN=00017
                  SENSEG NAME=S0000,PARENT=0
                  SENSEG NAME=S1000,PARENT=S0000
                  SENSEG NAME=S1300,PARENT=S1000
                  SENSEG NAME=S3000,PARENT=S0000
                  SENSEG NAME=S5000,PARENT=S0000
                  SENSEG NAME=S5100,PARENT=S5000
                  PSBGEN LANG=ASSEM,PSBNAME=EKYCCU00,                          C
                        IOEROPN=(451,WTOR),CMPAT=YES
                  END
EKYC095I SQL STATEMENT FOR PRID PR0000, TABLE <DB2PROD."T0000">:
SELECT "K0000", "C0000" FROM DB2PROD."T0000" ORDER BY "K0000"

EKYC095I SQL STATEMENT FOR PRID PR1000, TABLE <DB2PROD."T1000">:
         SELECT "K0000", "K1000", "C1000", "C1300" FROM
            DB2PROD."T1000" ORDER BY "K0000", "K1000"

EKYC095I SQL STATEMENT FOR PRID PR3000, TABLE <DB2PROD."T3000">:
         SELECT "K0000", "K3000", "C3000", "C3000IN1", "C3000IN2",
            "C3000DC1", "C3000DC2", "C3000TMS" FROM DB2PROD."T3000"
            ORDER BY "K0000", "K3000"

EKYC095I SQL STATEMENT FOR PRID PR5100A, TABLE <DB2PROD."T5100A">:
         SELECT "K0000", "K5000", "K5001", "K5002", "K5003", "K5100",
            "C5100VG" FROM DB2PROD."T5100A"  ORDER BY "K0000", "K5000",
            "K5001", "K5002", "K5003", "K5100"
```

Figure 392. Sample CCU Report: Direct Technique with the READONLY Keyword (Part 1 of 3)

```
EKYC095I SQL STATEMENT FOR PRID PR5100B, TABLE <DB2PROD."T5100B">:
         SELECT "K0000", "K5000", "K5001", "K5002", "K5003", "K5100",
            "C5100", "C5100TIM", "K5101" FROM DB2PROD."T5100B" ORDER BY
            "K0000", "K5000", "K5001", "K5002", "K5003", "K5100",
            "K5101"
EKYC002I CONTROL DATA SET SUCCESSFULLY CREATED.
TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING DIRECT TECHNIQUE, READONLY.
         DPROP SYSTEM NAME=DPROP001 TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS

EKYC003I INITIALIZATION PHASE ENDED NORMALLY.
==================================================================================================================
DPROP120   PGM=EKYC300X   JOB/STEP=JOB1     G        READ       DATE=93.001 TIME=12.00.00   DDN=CCUPRINT   PAGE=       1
 EKYX116I DPROP ACCESSING THE DPRMASTER ROW OF DPROP SYSTEM=DPROP001
 EKYC300I READ AND COMPARE PHASE STARTED
 EKYC815I CONTROL DATA SET SPECIFICATIONS:
         TIMESTAMP=1993-01-01-12.00.00.000000, PROCESSING DIRECT TECHNIQUE, READONLY.
         DPROP SYSTEM NAME=DPROP001 TOKEN=123456789012345, PROPAGATION=SYNCHRONOUS
EKYC828I THE DPROP DIRECTORY INDICATES ERROPT(PROPAGATION FAILURE)=IGNORE
OR A STATUS OTHER THAN 'ACTIVE' FOR THE FOLLOWING PRIDS:

         PRID     STATUS    ERROPT  SEGMENT  DB2 TABLE NAME
         -------- --------- ------- -------- --------------------------------------------------
         PR0000   ACTIVE    IGNORE  S0000    DB2PROD."T0000"
         PR5100A  INACTIVE  BACKOUT S5100A   DB2PROD."T5100A"
         PR5100B  SUSPENDED IGNORE  S5100B   DB2PROD."T5100B"
EKYC816I THE FOLLOWING IMS FIELD AND DB2 COLUMN DEFINITIONS MIGHT CAUSE DATA COMPARE PROBLEMS IN THE CCU.
FOR MORE INFORMATION, SEE 'NOTES' FOLLOWING THIS LIST.

         PRID     SEGMENT  FIELD    DATATYPE DB2 COLUMN NAME    DATATYPE NULLS  DB2 TABLE NAME                      NOTE
         -------- -------- -------- -------- ------------------ -------- ------ ----------------------------------- ----
         PR3000   S3000    F3000IN1 B        C3000IN1           SMALLINT YES    DB2PROD."T3000"                        2
                           F3000IN2 B        C3000IN2           INTEGER  YES                                           2
                           F3000IN3 B        C3000DC1           DECIMAL  YES                                           2
                           F3000PK  PACKED   C3000DC2           DECIMAL  W.DEF.                                        5
                           F3000TMS TIMESTMP C3000TMS           TIMESTMP W.DEF.                                        3
         PR5100A  S5100A   F5100LVG LONGVARG C5100VG            VARG     W.DEF. DB2PROD."T5100A"                       1
         PR5100B  S5100B   F5100TIM TIME     C5100TIM           TIME     W.DEF. DB2PROD."T5100B"                       4

         NOTES:  1) TRAILING BLANK CHARACTERS (X'40') OF THE FIELD AND COLUMN DATA WILL BE TRUNCATED BEFORE COMPARISON.
                 2) BLANK OR ZERO FIELD DATA COMPARED WITH DB2 COLUMN 'NULLS' WILL NOT BE CONSIDERED AS DATA MISMATCH.
                 3) TIMESTAMPS HAVE DIFFERENT LENGTH; ONLY THE COMMON TIMESTAMP DATA VALUE WILL BE USED FOR COMPARISON.
                 4) FIELD/COLUMN WILL NOT BE CHECKED BECAUSE THE DB2 COLUMN DEFINITION IS 'NOT NULL WITH DEFAULT'.
                 5) SCALE FACTORS ARE DIFFERENT; ONLY THE COMMON PART OF THE SCALE VALUE WILL BE USED FOR COMPARISON.
EKYC935I MISMATCH BETWEEN PROPAGATED IMS DATA AND DB2 TABLE <DB2PROD."T5100B">

         IMS KEY/ID FIELDS TO SEGMENT S5100
          KEY FIELD K0000    IN SEGMENT S0000       +0000   F2F2F1F6 F0F5                        *221605          *
          KEY FIELD K5000    IN SEGMENT S5000       +0000   F0F5                                 *05              *
          ID  FIELD F5000    IN SEGMENT S5000       +0000   F0F5                                 *05              *
          ID  FIELD F5001    IN SEGMENT S5000       +0000   F0F540                               *05              *
          ID  FIELD F5002    IN SEGMENT S5000       +0000   00000000 0000500C                    *......&.        *
          KEY FIELD K5100    IN SEGMENT S5100       +0000   F0F5                                 *05              *
              FIELD F5101    IN SEGMENT S5100B      MISSING

         DB2 PRIMARY KEY
          KEY COLUMN <K0000>                        +0000   F2F2F1F6 F0F5                        *221605          *
          KEY COLUMN <K5000>                        +0000   F0F5                                 *05              *
          KEY COLUMN <K5001>                        +0000   F0F5                                 *05              *
          KEY COLUMN <K5002>                        +0000   F0F540                               *05              *
          KEY COLUMN <K5003>                        +0000   00000000 0000500C                    *......&.        *
          KEY COLUMN <K5100>                        +0000   F0F5                                 *05              *
          KEY COLUMN <K5101>                        +0000   0001                                 *..              *
```

Figure 392. Sample CCU Report: Direct Technique with the READONLY Keyword (Part 2 of 3)

```
        WITHOUT CORRESPONDING OCCURRENCE OF INTERNAL IMS SEGMENT S5100B

         ** GENERATED REPAIR STATEMENT NUMBER:              1 **

-----------------------------------------------------------------------------------------------------------------------


EKYC310I STATISTICS FOR RETRIEVED IMS AND DB2 DATA:

         SEGMENT       RETRIEVED CHECK PRID     SEGMENT ROLE  CONVERTED SEGM                            DATA MISMATCHES
                                       MAPPING CASE/OPTIONS   RETRIEVED ROWS DB2 TABLE NAME
         ---------------------- ----- ---------------------- -------------- ------------------------------------------
         S0000                 4 YES   PR0000   ENTITY                    4                                          0
                                       MC1                                 4 DB2PROD."T0000"
                                 USED  PR5100A  PARENT
                                 USED  PR5100B  PARENT
                                 ----- ---------------------- -------------- ------------------------------------------
         S1000                 4 YES   PR1000   ENTITY                    4                                          0
                                       MC2                                 4 DB2PROD."T1000"
                                 ----- ---------------------- -------------- ------------------------------------------
         S1300                 2 YES   PR1000   EXTENSION                 4                                          0
                                       MC2                                 - DB2PROD."T1000"
                                 ----- ---------------------- -------------- ------------------------------------------
         S3000                 9 YES   PR3000   ENTITY                    9                                          0
                                       MC1                                 9 DB2PROD."T3000"
                                 ----- ---------------------- -------------- ------------------------------------------
         S5000                 4 USED  PR5100A  PARENT
                                 USED  PR5100B  PARENT
                                 ----- ---------------------- -------------- ------------------------------------------
         S5100                 6 YES   PR5100A  CONTAINING               10                                          0
                                       MC3 / PATHDATA                     10 DB2PROD."T5100A"
                                 YES   PR5100B  CONTAINING                6                                          1
                                       MC3 / PATHDATA                     7 DB2PROD."T5100B"
                                 ----- ---------------------- -------------- ------------------------------------------

EKYC303I 1 DATA INCONSISTENCIES LISTED WITHIN THIS RUN

EKYC304I READ AND COMPARE PHASE ENDED AT END OF DATA.
```

Figure 392. Sample CCU Report: Direct Technique with the READONLY Keyword (Part 3 of 3)

Chapter 20. CCU Front End

The CCU front end ensures the correct execution of all internal functions of the CCU front end. Execute it after installing the IMS DPROP system or systems and after defining your propagation requests.

The CCU front end consists of a REXX procedure, and a number of ISPF panels, skeletons and messages. You call it by entering a command on the ISPF command line, or by using the ISPF IMS DPROP Utility Selection panels.

The CCU front end runs under ISPF V3.2 and TSO/E V2.2 and later releases. It's purpose is to generate JCL for MVS/SP™ JES2; however, any release of JES2 that is supported by the MVS/SP Base Control Program is supported.

You can use the CCU front end to:

1. Generate JCL streams required to submit the CCU. The generation is based on ISPF skeleton JCL and input parameters you provide on ISPF panels.
2. Store the generated JCL stream in a library of your choice.
3. Retrieve CCU-related information from the IMS DPROP directory.

The following topics provide additional information:

- "How to Start the CCU Front End "
- "Requesting Help within the Application" on page 527
- "Panel Flow " on page 527
- "Panels for the CHECK control statement " on page 547
- "Panel EKYCP12E—Example of SEG Operand List" on page 552
- "IMS Database Allocation " on page 553

How to Start the CCU Front End

Start the CCU front end from the ISPF Primary Option Panels. Select option **6** enter TSO commands. Then from the TSO Command Processor screen, you either:

- Enter an EXEC command that takes you directly into the CCU front end
- Enter an EXEC command that takes you to the IMS DPROP utilities panels.

How to Start CCU Front End Using the Direct ISPF Method

To start the CCU front end directly, enter the following command on the ISPF command line:

```
EXEC ' prefix.SEKYCLIB(EKYCL00X)'
```

You cannot pass a parameter to the application when you use the direct method. However, you must specify the high-level qualifier for the IMS DPROP ISPF libraries in the REXX procedure, EKYCL00X. Edit EKYCL00X to specify the high-level qualifier in the variable name ISPFHLQ.

The CCU front end branches to the setup panels where you are prompted to enter parameters for your IMS DPROP system environment. See "Setup Panels " on page 530 for discussion on the setup panels.

How to Start CCU Front End Using the ISPF Utilities Panels

To use the ISPF panels to start the CCU front end, enter the following command on the ISPF command line:

```
EXEC ' prefix.SEKYCLIB(EKYNAPPL)' 'parm'
```

where:

prefix is all the high-level qualifiers to the left of SEKYCLIB

parm is the high-level qualifier of the IMS DPROP Release 2 data sets. It can be up to 26 characters in length, and it must be enclosed in quotes.

The next panel you see is EKYNPM0E, as shown in Figure 393.

```
EKYNPM0E ---------------- DPROP Utilities Applications ------------------------
  Command ===>
  @@@@@@@@    @@@@@@@@    @@@@@@@@      @@@@@    @@@@@@@@          @@@@@      @@@
  @@@@@@@@@   @@@@@@@@@   @@@@@@@@@    @@@@@@@   @@@@@@@@@        @@   @@    @@@@
  @@     @@   @@     @@   @@     @@   @@     @@   @@     @@            @@       @@
  @@     @@   @@     @@   @@     @@   @@     @@   @@     @@            @@       @@
  @@     @@   @@@@@@@@@   @@@@@@@@@   @@     @@   @@@@@@@@            @@       @@
  @@     @@   @@@@@@@@    @@@@@@@@    @@     @@   @@@@@@@@            @@       @@
  @@@@@@@@@   @@          @@    @@    @@@@@@@@   @@                @@  @@  @@  @@
  @@@@@@@@    @@          @@     @@    @@@@@    @@                @@@@@@@  @@  @@

              @   @  @@@@@  @@@  @      @@@  @@@@@  @@@  @@@@@   @@@@
              @   @    @     @   @       @     @     @   @      @
              @   @    @     @   @       @     @     @   @@@@   @@@@@
              @   @    @     @   @       @     @     @   @          @
               @@@     @    @@@  @@@@@  @@@    @    @@@  @@@@@  @@@@

    5685-124 (C) COPYRIGHT IBM(*) CORPORATION 1989,1994. All rights reserved.
           US Government Users Restricted Rights - Use, duplication or
        disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
                (*) - Trademark of International Business Machines.

            Enter END command to return or press <Enter> to continue.
```

Figure 393. IMS DPROP Utilities Application Entry Panel

Press the End key to exit the CCU front end.

Press enter to take you to panel EKYNPM1E, shown in Figure 394.

```
EKYNPM1E ------------------ DPROP Utility Selection ------------------
Command ===>
                                                        Application ID: (id)

 Enter the name of the DPROP Utilities Application:

      DPROP Utility ===>              ( CCU   - Consistency Check Utility )
                                      ( DLU   - DL/I Load Utilities       )
                                      ( MVGIN - MVG Input Tables Utility  )


    The DPROP utility applications have the ability to access the DPROP
    system definitions made during DPROP installation.
    If you want the invoked DPROP utility application to make use of this
    feature, enter the DPROP system name and the name of the library
    containing the customized DPROP installation job members:

      DPROP System  ===>                    ( * for DPROP system overview )
      Job Library   ===>


 Enter End command to return or press <Enter> to continue.
```

Figure 394. IMS DPROP Utilities Application Panel EKYNPM1E

Enter the name of the IMS DPROP utility application you want to call.

If you know the IMS DPROP system name and the name of the library containing the customized IMS DPROP installation job members for this IMS DPROP system, you can enter them in the appropriate fields.

If you know only the name of the job library, enter an asterisk (*) in the IMS DPROP System field to see the IMS DPROP systems using this job library. (You will see the pop-up panel shown in Figure 395.) The job library name should be the library name as entered during IMS DPROP installation.

When you press Enter, the application reads the definitions for this IMS DPROP system as entered in the IMS DPROP installation application, stores the definitions in the ISPF shared pool, and calls the IMS DPROP utility application you selected. The called IMS DPROP utility application tries to access the definitions from the ISPF shared pool. When they are present, you do not have to enter the definitions assigned to a IMS DPROP system.

The IMS DPROP system and the job library you enter are stored in the ISPF profile and are shown the next time you call the IMS DPROP Utilities Application.

If you know neither the IMS DPROP system name nor the name of the job library, blank out both fields and press Enter to call the IMS DPROP utility application you selected. The IMS DPROP utility application then displays the setup panel, EKYCP20E. See “Setup Panels ” on page 530 for a description of the setup panels.

```
EKYNPM1E ------------------ DPROP Utility Selection ---------------------------
Command ===>
                                                          Application ID: (id)

 Enter the name of the DPROP utility application you want perform:

      DPROP Utility ===>              ( CCU   - Consistency Check Utility )
                                      ( DLU   - DL/I Load Utilities       )
                                      ( MVGIN - MVG Input Tables Utility  )


    The DPROP utility applications have the ability to access the DPROP
    system definitions made during DPROP installation.
    If you want the invoked DPROP utility application to make use of this
    feature, ______________DPROP System Overview (Panel nn of nn)___________
    containin |  EKYNPM2E ==>                                              |
              |                                                            |
      DPROP S |  DPROP System Name : (name)     System Number:   (num)     |
      Job Lib |  DPROP System Title: (title)                               |
              |  DPROP Environment : (name)                                |
              |                                                            |
Enter End com |  Enter a minus sign (-) to go back one DPROP system or     |
              |  press <Enter> to continue or enter END command to  return.|
              |____________________________________________________________|
```

Figure 395. IMS DPROP System Overview Pop-up Panel

Calling the CCU Front End for the First Time

You use the panel in Figure 396 on page 526 the first time you call the CCU front end.

```
EKYCP01E ----- DPROP Data Consistency Check Utility - First Call -------------
Command ===>



       DPROP V1R2 - CCU Job Control Generation Application


       On the  first call to this procedure,  you should review the CCU
       Job Control Default values  and adapt them  to your  environment.

       Change the presented default values to values you need for your
       DPROP Environment and save them in your ISPF profile by pressing
       Enter on each screen.









Press: Enter to set the defaults                              End key to exit
```

Figure 396. CCU First Call Panel: EKYCP01E

Press the End key to exit the CCU front end.

Press the Enter key to branch to panel EKYCP20E to set the CCU initial defaults (see page 531). After you set the CCU defaults, continue with panel EKYCP00E to select CCU options, as described on page 529.

Calling the CCU Front End when Parameters Have Changed

You can change any default parameters in the CCU front end. After a change, you can force all users calling the application to verify their current CCU defaults in the setup panels. To do so, use the SEKYCLIB exec with the EKYCL00X procedure to change the value of the variable PROCMOD as instructed on the screen.

You use panel EKYCP02E, shown in Figure 397, when you are forced to verify the CCU default allocations.

```
EKYCP02E -- DPROP Data Consistency Check Utility - Modify Defaults -----------
Command ===>



       DPROP V1R2 - CCU Job Control Generation Application


       The procedure has changed, or default values are incompatible -
       adapt your CCU job control defaults.

       By pressing the Enter key you see the panels to modify the
       default job control values used for the job control generation.
       Query the presented default values,  make changes where necessary,
       and press the Enter key to save the values in your ISPF
       profile.





Press: Enter to modify the defaults                           End key to exit
```

Figure 397. CCU Modify Defaults Panel: EDKCP02E

Press the End key to exit the CCU front end.

Press the Enter key to branch to panel EKYCP20E to verify and modify the CCU defaults (see page 531). After you have completed setting the CCU default allocations, you continue with panel EKYCP00E (described on page 529).

Requesting Help within the Application

You can request the help panels either:

- From the command line, press the Help key to view a Help selection panel. From there, you can select help information on various topics.
- From a panel input field, press the Help key; you see the help panel for that input field. From there, you can press the UP key to get the Help Selection panel.

Tutorial pages with expanded selection possibilities are available from every dialog panel.

Panel Flow

This information describes the CCU front end panel flow, and lists the panel ID's and locations in this topic where you will find them.

Figure 398 on page 528 illustrates the panel flow. The sequence of the panels is:

IMS DPROP Utilities Application Entry
Use the ISPF panels to select either the CCU, DLU, or MVGIN utility, or use the direct method to go directly to the utility.

The IMS DPROP CCU Selection Panel
Select the following CCU activities from this main panel, shown in Figure 399 on page 530.

Setup Enter and verify CCU default values, which are saved in your ISPF profile. If you use the direct ISPF method to enter the CCU, you do not see the setup panels unless you do not specify the IMS DPROP system name or the library. You must specify default values before you can start generating JCL to run the CCU. When you next call the CCU front end, you can restore any incorrect default values from the ISPF profile by pressing the End key.

Generate or Gensave
Enter the input statements required to generate a job control stream. Use either the GENERATE or GENSAVE option on the main selection panel.

IMS DPROP merges your data with skeletal JCL data. The completed JCL data is left in a temporary data set, or copied to a data set of your choice. The data set containing the generated JCL is displayed in edit mode. You can then either submit the JCL or save it.

Query Search for specific information in the IMS DPROP directory, and browse the information.

The CCU supports the redisplaying of any of the setup panels during the job control generation. You can set this option in the Modify Defaults Panel, EKYCO20E.

To make the panel names visible on each panel, type PANELID ON on the command line of any panel. To remove the panel names, type PANELID OFF.

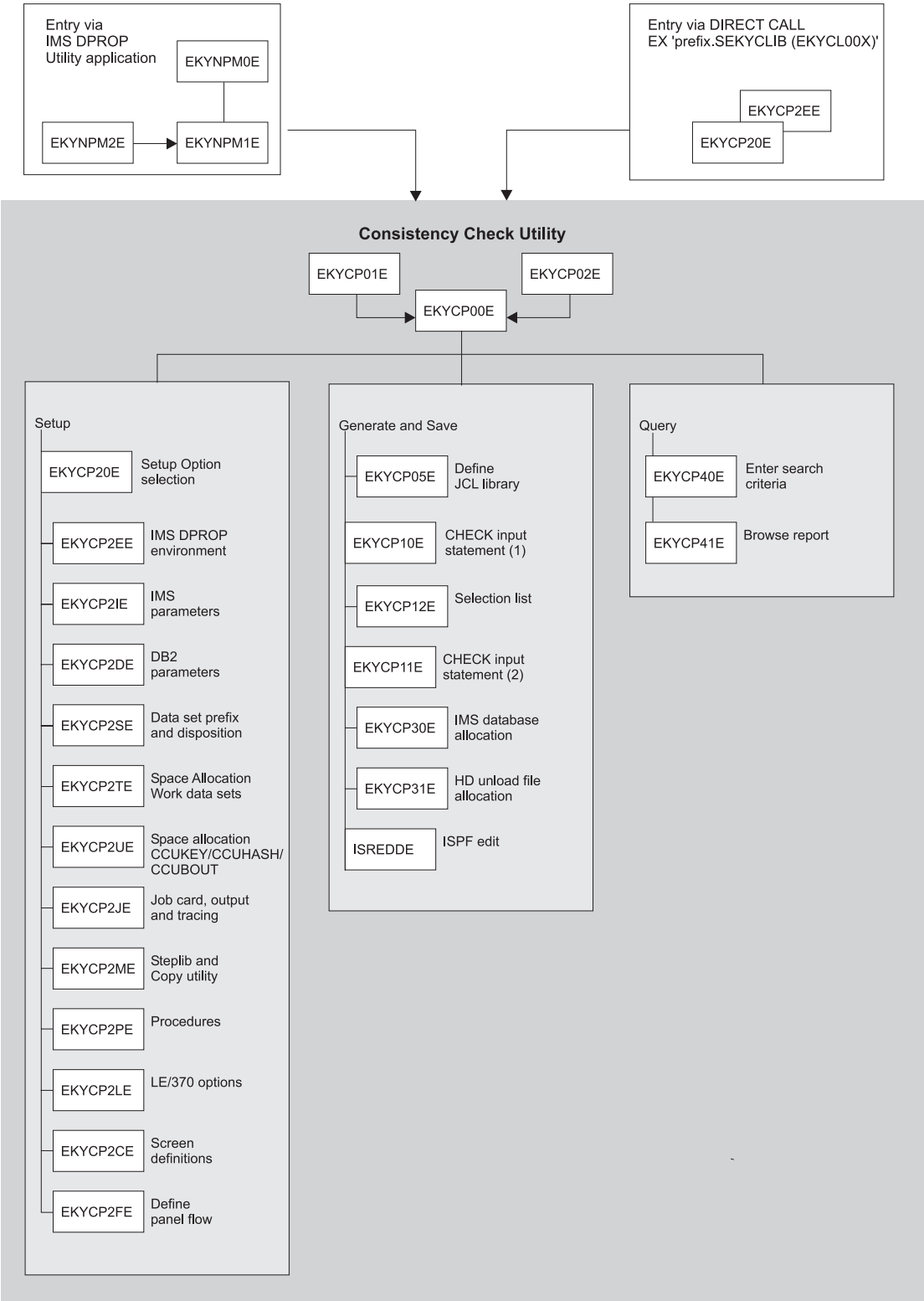

Figure 398. CCU Front End Panel Flow

Table 42 lists the CCU panel IDs, their titles, and the page numbers where they are described.

Table 42. CCU Application Panel Identifiers

Panel ID	Panel Title	Page
EKYCP00E	IMS DPROP Data Consistency Check Utility	529
EKYCP01E	IMS DPROP Data Consistency Check Utility - First Call	525
EKYCP02E	IMS DPROP Data Consistency Check Utility - Modify Defaults	526
EKYCP05E	Save JCL in Library	548
EKYCP10E	Enter CCU Check Command (1 of 2)	549
EKYCP11E	Enter CCU Check Command (2 of 2)	550
EKYCP12E	Select Items for Operand xxxxxxxx	553
EKYCP20E	Modify Defaults for System xxxxxxxx	531
EKYCP2CE	Modify Screen Setting	545
EKYCP2DE	Modify DB2 Parameters	535
EKYCP2EE	Enter/View IMS DPROP Environment Parameters	532
EKYCP2FE	Define Panel Flow	546
EKYCP2IE	Modify IMS Parameters	533
EKYCP2JE	Modify Job Parameters	540
EKYCP2LE	Modify AD/Cycle® LE/370 Run Time Options	545
EKYCP2ME	Modify Steplib and Copy Utility Parameters	542
EKYCP2PE	Modify Procedure Names	544
EKYCP2SE	Modify Data Set Prefix and Disposition	536
EKYCP2TE	Modify Data Set Allocation Work Data Sets	538
EKYCP2UE	Modify Data Set Alloc CCUKEY, CCUHASH, CCUDBOUT	539
EKYCP30E	Enter IMS Data Allocation Statements	554
EKYCP31E	Enter HD Unload File Name	555
EKYCP40E	Query IMS DPROP Directory	556
EKYCP41E	Query Report	558

Main Selection Panel

Use the main selection panel to indicate the CCU option you want to use. Enter an option in the command field, or move the cursor to an option and press Enter. To request more information, press Help. To exit the CCU front end, press End. Figure 399 on page 530 shows the Main CCU Selection Panel, EKYCP00E.

```
EKYCP00E ---------- DPROP Data Consistency Check Utility ---------------------
Selection ===>

                                                        DPROP System - xxxxxxxx
                                                              Userid -  xxxxxxx
Select one of the following options:                            Time -    xxxxx

   blank - GENERATE - Generate and Edit Job Control
   S     - GENSAVE  - Generate, Save, and Edit Job Control
   D     - SETUP    - Modify Job Control Generation Default Values
   Q     - QUERY    - Query DPROP Directory for IMS DPROP Information









Press: End to exit         Help for more information         Enter to continue
```

Figure 399. EKYCP00E: CCU Main Selection Panel

The main selection panel options are:

blank (GENERATE)
Generate a CCU job stream. Store it in a temporary data set from which you can edit or submit the JCL.

S (GENSAVE)
Generate a CCU job stream. Save it in a job control library of your choice from which you can edit or submit the JCL.

D (SETUP)
Verify and modify the CCU default allocations and save them in your ISPF profile.

Q (QUERY)
Query the DPROP directory for CCU-related information. This option runs only when DB2 is available.

Table 43 lists the panels used by each CCU option in panel EKYCP00E.

Table 43. Selections on Panel EKYCP00E

Option	Takes you to Panel	Described on Page
blank – Generate	EKYCP10E	549
S – Gensave	EKYCP05E	548
D – Setup	EKYCP20E	531
Q – Query	EKYCP40E	556

Setup Panels

You use the setup panels to modify defaults for the various environments and parameters used by the CCU front end. If any option in the panel is preceded by an asterisk, you must select that option and complete the dialog before you can start generating CCU job control language.

Figure 400 shows panel EKYCP20E, which is a menu from which to select the area whose defaults you want to modify.

```
EKYCP20E ---------- Modify Defaults for System xxxxxxxx ----------------------
Selection ===>

Select one of the following options to modify default values.

  *0 - DPROP Environment            (System, Propagation, Planname, DB2 SSID)
   1 - IMS Environment              (PSB, IMS Libraries, Database Allocation)
   2 - DB2 Environment              (SSM Identification, Subsystem Library  )
   3 - Data Set Allocation  I       (DS Prefix, Space For Work Data Sets    )
   4 - Data Set Allocation II       (CCUKEYn, CCUHASHn, CCUDBOUT Data Sets  )
   5 - Job Card and Trace           (Job Card, Output Class, Trace Level    )
   6 - Steplib and Utility          (Steplib, RESLIB, Userlib, Copy Utility )
   7 - Procedure Names              (IMS Batch, BMP, SORT, TSO-TMP          )
   8 - AD/Cycle LE/370              (LE/370 Runtime Options                 )

   C - Define Screen                (Command Line Placement, Panel Colors   )
   F - Define Panel Flow            (Redisplay Setup Panel in JCL Generation)

  *) needs to be selected


Press: End to exit         Help for more information         Enter to continue
```

Figure 400. EKYCP20E: CCU Modify System Defaults Panel

Enter an option in the command field, or move the cursor to the input field on the left of an option and press the Enter key.

To exit the setup panel press the End key, you then return to the main selection panel, EKYCP00E, described on page 529.

Table 44 lists the panels displayed based on the options you select in panel EKYCP20E.

Table 44. Selections on panel EKYCP20E

Option	Takes You to Panel	Described on Page
0 – IMS DPROP Environment	EKYCP2EE	532
1 – IMS Environment	EKYCP2IE	533
2 – DB2 Environment	EKYCP2DE	535
3 – Data Set Allocation 1	EKYCP2SE and EKYCP2TE	536 and 538
4 – Data Set Allocation 2	EKYCP2UE	539
5 – Job Card and Trace	EKYCP2JE	540
6 – Steplib and Utility	EKYCP2ME	542
7 – Procedure Names	EKYCP2PE	544
8 – AD/Cycle LE/370	EKYCP2LE	545
C – Define Screen	EKYCP2CE	545
F – Define Panel Flow	EKYCP2FE	546

Panel EKYCP2EE—The IMS DPROP Environment

Option **0** in setup selection panel EKYCP20E, takes you to panel EKYCP2EE, shown in Figure 401 on page 532. Use panel EKYCP2EE to define system definitions and DB2 and SQL identifiers used in the IMS DPROP environment.

If you entered the CCU front end through the IMS DPROP Utilities ISPF panels and specified a IMS DPROP system name and its installation library, all fields on this panel are protected and cannot be modified. Otherwise, you must verify or modify the fields.

If you want to accept the parameters displayed in this panel, press Enter. If you want to modify the parameters, type in the appropriate values.

After you have entered the information press:
- ENTER to commit the changes and return to panel EKYCP20E
- END to undo the changes
- Help to request more information

```
EKYCP2EE ---------- Enter/View DPROP Environment Parameters ------------------
Command ===>

Enter the DPROP System Definitions you want to generate JCL for:
   DPROP System Name    ===>
   Propagation          ===>              ( Synchronous or Asynchronous )
   Directory Qualifier  ===>

Enter the DB2 Subsystem Identification for this DPROP system:
   DB2 SubSystem ID     ===>

Enter the SQL String Delimiter that was defined for this DB2 System:
   SQL String Delimiter ===>             ( ' = Apostrophe
                                           " = Quotation Mark
                                           D = Default        )

Enter the DB2 Plan Name that was bound for the CCU for this DPROP System:
   DB2 Plan Name        ===>



Press Enter to commit changes or enter End command to return.
```

Figure 401. Panel EKYCP2OE: Set Up the IMS DPROP Environment

The parameters in panel EKYCP2EE are:

IMS DPROP System Name
The name of the IMS DPROP system for which you want to generate JCL. The IMS DPROP system name is a string of up to 8 bytes, starting with an alphabetic character.

Propagation
The type of propagation you use, either synchronous or asynchronous.

Directory Qualifier
The DB2 table qualifier generated for the IMS DPROP system.

DB2 Subsystem ID
The name of the DB2 subsystem under which the IMS DPROP system is running. The DB2 subsystem ID is a string of up to 4 bytes. It must be alphanumeric and start with an alphabetic character. It is used in all CCU job steps that require connection to DB2.

SQL String Delimiter
The SQL string delimiter defined for the DB2 subsystem. This can be either a quotation mark, an apostrophe, or a default value.

CCU Plan Name
The name of the DB2 application plan that was bound to allow the CCU access

to the IMS DPROP directory tables of the IMS DPROP system. The CCU plan name is a string of up to 8 bytes. It must be alphanumeric and start with an alphabetic character.

Job steps that require connection to DB2 must use the CCU plan name. Examples of these job steps are:

- CCU initialization phase
- DB2 read phase
- IMS read and error location phase, if running if an IMS batch region

Both the CCU initialization phase and the DB2 read phase run in TSO-TMP regions.

Use panel EKYCP2DE, described on page 535. to enter the SSM identifier and subsystem library, which are both used if you submit the CCU in a BMP region.

Panel EKYCP2IE–The IMS Environment

Option **1** in setup selection panel EKYCP20E, takes you to panel EKYCP2IE, shown in Figure 402. Use EKYCP2IE to define IMS parameters for the PSB load module, IMS libraries, and allocation method for the IMS database.

```
EKYCP2IE ---------------- Modify IMS Parameters -----------------------------
Command ===>

Enter Name and Load Library of the PSB load module generated by the CCU:
   PSB Load Module Name  ===>
   PSB Load Library      ===>

Default DBPCB values to be used by CCU to generate the PSB load module:
   PCB Processing Option ===>            (e.g. G, GO, GE, etc)
   PCB Key Length        ===>            (blank for determination by CCU)

Enter following IMS libraries to be used in the CCU job control:
   Name of IMS DBDLIB    ===>
   Name of IMS ACBLIB1   ===>
   Name of IMS MACLIB    ===>

The IMS database to be processed by the CCU can be allocated either in the
generated job control, or, alternatively, using the IMS dynamic allocation
exit (DFSMDA). Select type to be used for the CCU job control:
   Database Allocation   ===>            (JCL or DYN)

Press Enter to commit changes or enter End command to return.
```

Figure 402. Panel EKYCP2IE: Modify IMS Parameters

The parameters in panel EKYCP2IE are:

PSB Load Module Name

The name of the PSB load library that contains the member that results from link editing the IMS DB-PCB source code.

The CCU uses your CHECK input statement and input from the IMS DPROP directory during the initialization phase to create IMS DB-PCB source code. Then it assembles and link-edits the source code to be a member of a PSB load library.

An old PSB with the same member name will be replaced by the new generated PSB load module. The CCU used the PSB name you enter as the default value when you enter the CHECK input command on panel EKYCP11E. There, you can overwrite it, as needed, for a subsequent call to the application.

The PSB load module name can be a string of up to 8 bytes. It must be alphanumeric, and start with an alphabetic character.

PSB Load Library
The name of the PSB load library that will contain the new PSB load module that resulted from the CCU initialization phase.

The PSB load library name is a string of up to 44 bytes, and it must conform to the naming conventions of MVS data sets.

PCB Processing Option
A value to indicate to IMS how you will access IMS data when using unqualified GET NEXT (GN) calls. Enter any valid combination of G, E, S, O, (T), and H. The *IMS/ESA Utilities Reference: System* explains these access codes.

The CCU requires a DB-PCB to communicate with IMS. The PROCOPT PROCOPT= operand of the CHECK command uses the PCB processing option when you generate a DB-PCB. You can override the PCB option, if needed, for subsequent calls to the application.

The CCU does not support IMS **GG** status codes. If you define a processing option in any combination with **T**, it does not suppress termination of the CCU when it receives an IMS **GG** status code.

PCB Key Length
The length of the longest concatenated key for a hierarchic path of sensitive segments used by the CCU within a current CHECK request. This value is needed to generate the PCB. Specify any value between 1 and 99999. If you leave this field blank, the CCU determines a value and uses it for the PSB load module that is being generated.

The PCB KEYLEN value you enter here is used as the default value for the CHECK input command on panel EKYCP11E. There, you can overwrite it, as needed, for a subsequent call to the application.

Name of IMS DBDLIB
The name of the IMS DBDLIB that contains the current valid IMS DBD load modules. It is a string of up to 44 bytes and must conform to the naming conventions for MVS data sets. The name is used by the CCU initialization phase, which needs access to the DBD load module for the IMS database whose data you want to check. The IMS read and error location phases need the name of the IMS DBDLIB that contains the DBD load module in the //IMS DD statement.

Name of IMS ACBLIB1
The name of the data set that contains the application control blocks (ACBs) of your own BMP programs. It must be an ACBLIB other than the primary IMSVS.ACBLIB, and it must be concatenated to it. The name is a string of up to 44 bytes and must conform to the naming conventions of MVS data sets.

You need the name of an IMS ACBLIB to build an ACB. If you submit the CCU in an IMS BMP region, you must do an ACBGEN to build the ACB for IMS.

Note that if you build the ACB in the generated ACBGEN step, you might replace an already existing member with the same name.

Name of IMS MACLIB
The name of the data set that contains the IMS macros required to compile a PSB. It is a string of up to 44 bytes and must conform to the naming conventions of MVS data sets.

Database Allocation

The type of IMS database allocation you use. Use DYN if you are allocating the IMS database dynamically, or enter JCL if you are allocating the IMS database from your JCL.

If you allocate IMS databases individually in your JCL, specify the IMS database's DD statements. When you enter JCL for the IMS database allocation, panel EKYCP30E asks you for the IMS database allocation DD statements. See page 554.

Enter the appropriate values and press either:
- ENTER to commit the changes and return to panel EKYCP20E
- END to undo the changes
- Help to request more information

Panel EKYCP2DE—The DB2 Environment

Option **2** in setup selection panel, EKYCP20E, takes you to panel EKYCP2DE, shown in Figure 403. Use panel EKYCP2DE to specify parameters used when you connect to DB2. To connect to DB2 you submit JCL by either:

- Using an RTT and entering a value for the SSM= parameter. You create an RTT and store it in the IMS.RESLIB by entering the SSM ID and subsystem library name.
- Allocating the DDITV02 and DDOTV02 DD statements. See the sample JCL in Figure 302 on page 426 and Figure 304 on page 432.
- Allowing the procedure initialization to find a DB2 application plan that corresponds to the application program name.

If you will submit the CCU in a BMP region and use a resource translation table (RTT), specify the SSM ID and the subsystem library name.

If you submit the CCU in a BMP region but do not use an RTT, you need at least three additional DB2 plans for the CCU. The DB2 plan names must be EKYC100X, EKYC300X and EKYC600X.

```
EKYCP2DE ---------------- Modify DB2 Parameters -----------------------------
Command ===>

Enter the name of an SSM Identifier, and its Subsystem Library, to be used
in an IMS BMP (Batch Message Processing) region, if you have planned to
submit the CCU in a BMP region using an RTT (Resource Translation Table),
or leave blank if not submitting the CCU in a BMP region using an RTT.

   SSM Identifier     ===>

   Optionally, enter the name of the Subsystem Library that contains the
   SSM Member.  If you enter a Subsystem Library, it will be allocated
   to the DFSESL DD statement.

   Subsystem Library  ===>

   For more details on SSM, RTT and DFSESL,
   see IMS System Definition Reference.



Press Enter to commit changes or enter End command to return.
```

Figure 403. Panel EKYCP2DE: DB2 Parameters

The parameters in panel EKYCP2DE are:

SSM Identifier
The 1- to 4-character suffix that you specify, together with the IMS identification of the currently assigned IMS system, to form the name of a member in the IMSVS.PROCLIB. For more information about the SSM identifier see *IMS System Definition Reference*.

Subsystem Library
Enter the name of the library that contains the SSM member. If the RTT you are using for the CCU resides in a library that is not in the JOBLIB/STEPLIB/LINKLIST concatenation, or if the library is not APF-authorized, IMS requires that you define the //DFSESL DD statement. The data set name in this field is allocated to the //DFSESL DD statement. If you leave this field blank, IMS searches the RTT in the JOBLIB/STEPLIB/LINKLIST concatenation.

Enter the appropriate values and press either:
- ENTER to commit the changes and return to panel EKYCP20E
- END to undo the changes
- Help to request more information

Note: The DB2 subsystem ID and the CCU plan name, which are both DB2 parameters, can be entered on panel EKYCP2EE. Its description is on page 532.

Panel EKYCP2SE and EKYCP2TE—Data Set Space Allocation

Option **3** in setup selection panel EKYCP20E, takes you to panels EKYCP2SE and EKYCP2TE, shown in Figure 404 and Figure 405 on page 538.

Use panel EKYCP2SE to define the disposition and prefix for data sets created in CCU job steps.

```
EKYCP2SE ----------- Modify Data Set Parameters (1 of 2) ----------------------
Command ===>

Enter the following parameters for the allocation of all CCU data sets:

   Data Set Disposition ===>             (PASS or CATLG)

   Data Set Qualifier   ===>
                              This data set qualifier will be enlarged by
                              a timestamp qualifier and the data set ddname
                              to make it unique across the MVS image.









Press Enter to commit changes and continue or enter End command to return
```

Figure 404. Panel DKYCP2SE: Modify Data Set Parameters (1 of 2)

The parameters used in panel EKYCP2SE are:

Data Set Disposition
Specifies whether or not you want to save a data set by cataloging it.

Enter CATLG if you want the data sets cataloged. If you enter CATLG, the data sets are cataloged after they are created, and stay in your system until you

delete them. An advantage of CATLG is you can restart a terminated CCU job step without restarting the entire CCU job stream. CATLG is recommended.

If you enter PASS, the data sets stay only as long as the CCU job stream is running. If a CCU job step terminates, you have to restart the whole CCU job stream. Exceptions to this are the generated repair data sets //CCUDERR and //CCUIERR, which are cataloged even if you specify PASS.

Data Set Qualifier

A data set qualifier to use as the prefix for the CCU data sets. The CCU adds to it the following identifiers:

- A timestamp, containing the character *D*, the day of the year and the time you are generating the CCU job control
- The name of the DD statement

For example, if you enter a value of *IMS DPROP.CCU*, and you are generating the CCU job control on January the 7th at 08:15 AM, then the data set name for the //CCUCDS data set will be:

```
DSN=IMS DPROP.CCU.D0070815.CCUCDS
```

Enter the appropriate values and press either:

- ENTER to commit the changes and continue on panel EKYCP2TE
- END to undo the changes
- Help to request more information

Use panel EKYCP2TE to choose either SMS-managed or non-SMS-managed data set allocation for the following data sets:

ASYSIN PSBGEN in initialization phase.

ASYSLIN PSBGEN in initialization phase.

CCUCDS Control data set, all phases.

CCUDERR Generated DB2 repair statements.

CCUHSUM1 Hash sum data set of IMS read phase.

CCUHSUM2 Hash sum data set of DB2 read phase.

CCUIERR Generated DL/I repair statements.

CCUMSMTC Mismatch indication records.

CCUSORTS Sort statements.

```
EKYCP2TE ----------- Modify Data Set Parameters (2 of 2) ---------------------
Command ===>

Enter the Space Allocation Parameters for following Data Sets: ASYSIN,
ASYSLIN, CCUCDS, CCUHSUM1, CCUHSUM2, CCUMSMTC, CCUDERR, CCUIERR, CCUSORTS.

   Space Units ===>               (CYL or TRK - optional for SMS data sets)
   Quantity    ===>               (e.g. 2,10  - optional for SMS data sets)

   For Non-SMS-Managed data sets:
      Device (Unit name) ===>               (e.g. 3380, SYSALLDA)
      Volser Number(s)   ===>

   For SMS-Managed data sets:
      Storage Class      ===>               (e.g. CLASS1, or blank for ACS)





Press Enter to commit changes or enter End command to return.
```

Figure 405. Panel EKYCP2TE: Modify Data Set Parameters (2 of 2)

The parameters in panel EKYCP2TE are:

Space Units
TRK or CYL, depending on whether you want to allocate the space quantity in tracks or cylinders.

Quantity
The quantity of space units to be used in primary and, if required, secondary allocation.

The space units and quantity fields plus a RLSE keyword result in the SPACE= parameter:

```
SPACE=( units,(quantity),RLSE)
```

For example, if you enter TRK in the Space Units field, and 1,2 in the Quantity field the SPACE parameter is:

```
SPACE=(TRK,(1,2),RLSE)
```

For non-SMS-managed data sets, enter the device and volser number.

Device (Unit name)
The number, type or group name of the device on which to allocate the data sets.

Device number specifies a particular device. It is a 3-character hexadecimal number for the device, such as 2A0.

Device type is an IBM-supplied name that identifies a device by its machine type and model, such as 3380 or 3400-5.

Group name requests a group of devices by a symbolic name. The group name is 1 to 8 alphanumeric characters, such as SYSALLDA.

Volser Number
The volumes on which to allocate the data sets. You can enter up to 4 volume serial numbers. If you enter more than one, separate them with a comma. Each volume serial number can be 1 to 6 alphanumeric, national ($, #, @), or special characters.

Storage Class
Specifies the name of a storage class to use to allocate the SMS-managed data sets. The storage class name is 1 to 8 characters and is defined by the storage administrator at your installation. If your installation uses ACS and you want the data sets allocated by ACS, leave the field blank.

Enter the appropriate values and press either:
- ENTER to commit the changes and return to panel EKYCP20E
- END to undo the changes
- Help to request more information

For more information on SMS, see*MVS/DFP™ Managing Non-VSAM Data Sets* and*MVS/DFP Access Method Services for the Integrated Catalog Facility*.

Panel EKYCP2UE—Data Set Space Allocation

Option **4** in setup selection panel EKYCP20E takes you to panel EKYCP2UE, as shown in Figure 406. Use EKYCP2UE to define Space Allocation parameters for the data sets CCUKEY1, CCUKEY2, CCUHASH1, CCUHASH2, and CCUDBOUT. See Table 37 on page 418 for space calculation rules.

```
EKYCP2UE ---------------- Modify Data Set Parameters -------------------------
Command ===>

For Non-SMS-Managed Data Sets:  Enter Space Units (TRK or CYL), Quantity,
    Device and Volser Number(s).  Leave Storage Class blank.
For SMS-Managed data sets:  Leave Device and Volser Number(s) blank.
    Optionally enter Space Units (TRK or CYL), Quantity, and Storage Class.

              -1-> Space Units       Device            Storage Class
              -2-> Quantity          Volser Number(s)
    CCUKEY1  ===>               ===> n/a          ===>
             ===>               ===>
    CCUKEY2  ===>               ===> n/a          ===>
             ===>               ===>
    CCUHASH1 ===>               ===>              ===>
             ===>               ===>
    CCUHASH2 ===>               ===>              ===>
             ===>               ===>
    CCUDBOUT ===>               ===>              ===>
             ===>               ===>

Press Enter to commit changes or enter End command to return.
```

Figure 406. Panel EKYCP2UE: Modify Data Set Parameters

The parameters in panel EKYCP2UE are:

CCUKEY1 and CCUKEY2
The space allocation and storage class. These files describe the VSAM KSDS data sets that contain the IMS segment concatenated keys, and DB2 row primary keys. The CCU creates the data sets during the IMS and DB2 read phases.

If the amount of data you are checking is large, you might decrease the time of the CCU run if you select different devices, volsers, or storage classes for each of the two files.

CCUHASH1 and CCUHASH2
The appropriate space allocation, volser, and storage class information. These files contain one record built from the hashed values of the data for every IMS segment occurrence or DB2 row. The data sets are created during the IMS and DB2 read phases when you use the hashing technique.

If the amount of data you are checking is large, you might decrease the time of the CCU run if you select different devices, volsers, or storage classes for each of the two files.

CCUDBOUT
The appropriate space allocation, volser, and storage class information. This file contains one mapped segment record in a CCU-internal format for every IMS segment type being checked. It is created by the IMS read phase if you are using an HD unload file as replacement to the IMS database.

For non-SMS-managed data sets, enter values for space units, quantity, device, and volser number or numbers.

Space Units
TRK or CYL, depending on whether you want to allocate the space quantity in tracks or cylinders.

Quantity
The quantity of space units to be used in primary and, if required, secondary allocation.

The space units and quantity fields together with an RLSE keyword results in the SPACE parameter. For example, if you specify CYL as the space unit and 10,20 as the quantity, the SPACE parameter is:

```
SPACE=(CYL,(10,20),RLSE)
```

Device
Enter the number, type or group name of the device on which to allocate the data sets.

Device number specifies a particular device. It is a 3-character hexadecimal number for the device, such as 2A0.

The device type is an IBM-supplied name that identifies a device by its machine type and model, such as 3380 or 3400-5.

Group name requests a group of devices by a symbolic name. The group name is 1 to 8 alphanumeric characters, such as SYSALLDA.

Volser Number(s)
Specify the volume on which to allocate the data sets. The volume serial number can be 1 to 6 alphanumeric, national ($, #, @), or special characters.

Storage Class
Specify the name of a storage class to use to allocate the SMS-managed data sets. The storage class name is 1 to 8 characters and is defined by the storage administrator at your installation. If your installation uses ACS and you want the data sets allocated by ACS, leave the field blank.

Enter the appropriate values and press either:
- ENTER to commit the changes and return to panel EKYCP20E
- END to undo the changes
- Help to request more information

Panel EKYCP2JE—Job Card, Output Classes, and Tracing Options

Option **5** in setup selection panel EKYCP20E, brings you to panel EKYCP2JE, as shown in Figure 407 on page 541. Use panel EKYCP2JE to specify the job card information to include with your JCL, and output and tracing options.

```
EKYCP2JE ------------------- Modify Job Parameters ---------------------------
Command ===>

Enter CCU Job Card Statements:
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>

Output: Enter Default Output Class for CCUPRINT and EKYTRACE Output:
   CCUPRINT    ===>          ( output class or DSN to write to a data set )
   EKYTRACE    ===>          ( output class or DSN to write to a data set )

Tracing: Enter Trace Debug Level and CCU Phase Name:
   Trace Level ===>          ( 0 - 127, e.g. 4,16,64.  0 = TRACE OFF )
   CCU Phase   ===>          ( INIT, READ, ERRORLOC or blank for all )

Press Enter to commit changes or enter End command to return.
```

Figure 407. Panel EKUCP2JE: Modify Job Parameters

The parameters in panel EKYCP2JE are:

Job Card Statements
Enter a job card statement to be used for the generated CCU job streams. There are 8 lines of input; you must fill in at least one line. If your linklist, SYS1.PARMLIB(LNKLSTnn), does not include the IMS RESLIB, DB2 LOADLIB, DXT LOADLIB or IMS DPROP RESLIB, you can add a JOBLIB statement.

Output
Define output classes for the CCUPRINT and EKYTRACE DD statements. Enter DSN to print the reports to a sequential data set instead of routing them to an output class.

CCUPRINT
Enter a 1-byte character output class to use for messages and reports the CCU generates. To print the generated messages and reports to a sequential data set, enter DSN with the following attributes:

Record format:
FBA

Record length:
121

Block size: 1210

EKYTRACE
Enter a 1-byte character output class to be used for the trace records generated by the CCU. The value you enter is effective only if you enter a trace level higher than 0. To print the generated messages and reports to a sequential data set, enter DSN with the following attributes:

Record format:
FBA

Record length:
133

Block size: 3990

Tracing

Specify the trace level and the CCU phase to trace.

Trace Level

Enter a value between 0 and 127. Trace level 64 is the preferred level for tracing the CCU. See Table 45 for a short list of applicable CCU trace levels.

Table 45. Applicable Trace Levels for CCU

Level	Trace Output
0	Do not trace
1	Provides an internal wraparound trace showing module entries and exits, including user exit routines
2	Provides an external trace of the “RUP changed IMS data” and the “HUP changed DB2 data.”
4	Provides an external trace of the calls to the RUP and to the HUP, with all interface information, including changed data and interface control blocks A level 4 trace also provides information on calls to Segment and Field exit routines, including the input and output of the exits
8	Provides an external trace of entries and returns for modules and major routines
16	Provides an external trace of relevant control block information at internally specified locations
64	Provides CCU information only, without the RUP and HUP control block information
127	Provides all available trace information

CCU Phase

Enter the symbolic CCU phase name to initiate a trace:

INIT Initialization phase

READ IMS and DB2 read phases of the hashing technique, or the read and compare phase of the direct technique

ERRORLOC Error location phase

blank All CCU phases

Enter the appropriate values and press either:
- ENTER to commit the changes and return to panel EKYCP20E
- END to undo the changes
- Help to request more information

Panel EKYCP2ME—Steplib And Copy Utility

Option **6** in setup selection panel EKYCP20E brings you to panel EKYCP2ME, shown in Figure 408 on page 543. Use panel EKYCP2ME to name libraries and a copy utility.

```
EKYCP2ME -------- Modify Steplib and Copy Utility Parameters -----------------
Command ===>

Enter the names of the following load libraries, used in the generated JCL:
   DPROP RESLIB              ===>
   IMS Load Library          ===>
   DB2 Load Library          ===>
   Additional Steplib(s)   1 ===>
     (e.g. for user exits) 2 ===>

Enter the name of the Utility Program  used for the JCL Copy process and
library where the program resides.  The program name you specify must be
compatible with SYSPRINT,  SYSIN,  SYSUT1, and SYSUT2 DD statements  (as
used by the IEBGENER utility).
   Copy Program Name         ===>
   Copy Program Load Library ===>





Press Enter to commit changes or enter End command to return.
```

Figure 408. Panel EKYCP2ME: Steplib and Copy Utility Parameters

The parameters in panel EKYCP2ME are:

CCU Program Load Library
Enter the name of the load library that contains the CCU load modules. It is the same library as the IMS DPROP RESLIB. The CCU uses this value in the //STEPLIB DD statement of the CCU job steps.

Additional Steplib(s)
The name of the library that contains your segment and field exits. Leave the field blank if you do not have any exits. The CCU adds this value to the //STEPLIB DD statement of the IMS and DB2 read phase of the hashing technique, the read and compare phase of the direct technique, and the error location phase.

IMS DPROP RESLIB
Enter the name of the load library that contains the authorized IMS DPROP modules. It is the same library as the CCU Program Load Library. The CCU uses this value in the //EKYRESLB DD statement of the CCU job steps.

Copy Program Name
Enter the name of a utility or self-written program, that is used to copy the generated job control language (JCL). The copy utility must be compatible with the IEBGENER utility. It is used to copy generated JCL from the temporary ISPF data set to a member of a job library you specify.

Copy Program Load Library
Enter the name of the load library that contains the copy program you named in the Copy Program Name parameter.

Enter the appropriate values and press either:
- ENTER to commit the changes and return to panel EKYCP20E
- END to undo the changes
- Help to request more information

Panel EKYCP2PE—Modify Procedure Names

Option **7** in setup selection panel EKYCP20E takes you to panel EKYCP2PE, shown in Figure 409. Use panel EKYCP2PE to name IMS procedures to initialize regions, sort the repair data sets, and submit jobs in a TSO-TMP region.

```
EKYCP2PE ---------------- Modify Procedure Names -----------------------------
Command ===>

Enter the Names of the following IMS Procedures to be used in the CCU job
control. They must reside in an active PROCLIB.
   Mixed IMS-DB2 Batch Region ===>
   Mixed IMS-DB2 BMP Region   ===>

Enter the Name of a SORT Procedure in an active PROCLIB:
   Sort Procedure             ===>

Enter the Name of a TSO-TMP (Terminal Monitor Processing) Program:
   TSO-TMP Program Name       ===>









Press Enter to commit changes or enter End command to return.
```

Figure 409. Panel EKYCP2PE: Modify Procedure Names

The parameters used in panel EKYCP2PE are:

Mixed IMS-DB2 Batch Region
Specify the name of a procedure that initializes an IMS batch region and allows it to connect to DB2. The procedure must exist in an active PROCLIB. This procedure is used by the IMS read phase of the hashing technique, the read and compare phase of the direct technique, and the error location phase. The default is DLIBATCH.

Mixed IMS-DB2 BMP Region
Specify the name of a procedure to initialize an IMS BMP region and allow it to connect to DB2. It must exist in an active PROCLIB. This procedure is used by the IMS read phase of the hashing technique, the read and compare phase of the direct technique, and the error location phase. The default is IMSBATCH.

Sort Procedure
Specify the name of a procedure that sorts the generated repair data sets, and resorts the mismatch indication records. The procedure is used after the compare phase of the hashing technique using an HD unload file. It must reside in an active PROCLIB. If you do not sort the generated repair data sets, and if you do not submit the CCU using an HD unload file, you can leave the SORT procedure blank.

The SORT procedure must contain all required SORT DD statements except //SORTIN, //SORTOUT, //SYSIN, AND //SYSOUT. which are generated by the CCU front end. The initial default is SORT.

TSO-TMP Program Name
Specify the name of a program to submit a job step using a TSO-TMP region. The default is IKJEFT01.

Enter the appropriate values and press either:
- ENTER to commit the changes and return to panel EKYCP20E

- END to undo the changes
- Help to request more information

Panel EKYCR2LE—AD/Cycle LE/370 Run Time Options

Option **8** in setup selection panel EKYCP20E takes you to panel EKYCP2LE, shown in Figure 410. Use this panel to define run time options for LE/370 if you have installed LE/370 and have IMS DPROP exit routines written in a high-level language supported by LE/370. The run time options are allocated to the //EKYLEOPT DD statement. For more information, see and IBM SAA AD/Cycle Language Environment/370 Programming Guide.

```
EKYCP2LE ----------- Modify AD/Cycle LE/370 Parameters -----------------------
Command ===>

If you've installed Language Environment/370 (LE/370) and you are
using user exits written in a high-level language supported by LE/370,
you may enter LE/370 Run Time Options below.
Enter either a data set containing the Run time Options or enter the Runtime
Options in the input lines, or leave all blank if you do not need them.

Enter Data Set Name Containing LE/370 Run Time Options:
    ===>

Enter LE/370 Run Time Options In The Following Input Lines:
    ===>
    ===>
    ===>





Press Enter to commit changes or enter End command to return.
```

Figure 410. Panel EKYCP2LE: LE/370 Parameters

The parameters used in panel EKYCP2LE are:

Data Set Name Containing LE/370 run time options
Enter the data set name that contains your LE/370 run time options. Leave blank if you enter the run time options in the input lines.

Define LE/370 Run Time Options instream
Enter the LE/370 run time options. Enter up to three lines of options, or leave blank if you entered a data set that contains run time options.

Enter the appropriate values and press either:
- ENTER to commit the changes and return to panel EKYCP20E
- END to undo the changes
- Help to request more information

Panel EKYCP2CE—Screen Definitions

Option **C** in setup selection panel EKYCP20E takes you to panel EKYCP2CE, shown in Figure 411 on page 546. On panel EKYCP2CE, you can modify the placement of the command line and set the colors for the CCU front end panels for graphic terminals.

```
EKYCP2CE ------------------ Modify Screen Setting ----------------------------
Command ===>

Command Line Placement    ===>         (ASIS or BOTTOM)

Select Colors, choose between WHITE, RED, BLUE, GREEN, TURQ, YELLOW, PINK
Press Enter to see the affect on the sample panel:

   Command Line                          ------------------------------
      Protected    High   ===>           Command ===> Input
      Unprotected  High   ===>
                                         This sample panel shows you
   Panel body                            the current setting.
      Protected    Low    ===>
      Protected    High   ===>             Input label ===> input
      Input Labels Low    ===>
      Unprotected  High   ===>                 This is a Special Text

      Special Text High   ===>           Press End when finished
                                         ------------------------------

Enter End command when finished or press Enter to redisplay screen.
```

Figure 411. Panel EKYCP2CE: Modify Screen Setting

The parameters used in panel EKYCR2CE are:

Command Line Placement
Changes the placement of the command line to:

ASIS Displays the command line as defined in the panel definition. The CCU front end defines the command line as the second line of each screen.

BOTTOM Displays the command line as the last line on the screen or as the last line above the split line.

The command line placement changes the next time you call the CCU front end.

Colors for Command Line and Panel Body
If you are working on a terminal that supports color setting, you can change the colors for the item types listed. Change colors and press Enter to see the effect on the sample panel.

Enter the appropriate values and press the End key to commit the changes and return to panel EKYCP20E.

Panel EKYCP2FE—Define Panel Flow

Option **F** in setup selection panel EKYCP20E gives you panel EKYCP2FE, as shown in Figure 412 on page 547. On this panel, you can define whether you want to display one or more of the setup panels while you are generating job control for the CCU.

If you select a panel, it prompts you to verify, or modify, the job control defaults for the *current* invocation of the procedure. If you change a value, it is not saved in the profile.

```
EKYCP2FE -------------------- Define Panel Flow ------------------------------
Command ===>

During generation of the job control, after you specified the CHECK input
statements, you can optionally re-display one or multiple of the following
setup panels to view or modify the corresponding job defaults.

Enter YES below to re-display the panel during job control generation.
Enter NO below to skip the panel during job control generation.

 DPROP Environment      (System, Propagation, Planname, DB2 SSID) ===>
 IMS Environment        (PSB, IMS Libraries, Database Allocation) ===>
 DB2 Environment        (SSM Identification, Subsystem Library  ) ===>
 Data Set Allocation  I (DS Prefix, Space For Work Data Sets    ) ===>
 Data Set Allocation II (CCUKEYn, CCUHASHn, CCUDBOUT Data Sets  ) ===>
 Job Card and Trace     (Job Card, Output Class, Trace Level    ) ===>
 Steplib and Utility    (Steplib, RESLIB, Userlib, Copy Utility ) ===>
 Procedure Names        (IMS Batch, BMP, SORT, TSO-TMP          ) ===>
 AD/Cycle LE/370        (LE/370 Runtime Options                 ) ===>


Press Enter to commit changes or enter End command to return.
```

Figure 412. Panel EKYCP2FE: Define Panel Flow

IMS DPROP, IMS, and DB2 Environment Setup Panels
Enter YES if you want to be prompted to verify or change input fields defining the IMS DPROP environment. Enter NO if you do not want to be prompted.

Data Set Allocation I
Enter YES if you want to be prompted to verify or change the data set prefix or space allocations for work data sets. If you specify YES, you see both EKYCP2SE and EKYCP2TE panels. Enter NO if you do not want to be prompted.

Data Set Allocation II
Enter YES if you want to be prompted to verify or change the Data Set Allocation parameters for CCUKEYn, CCUHASHn, CCUDBOUT. Enter NO if you do not want to be prompted.

Job Card and Trace
Enter YES if you want to be prompted to verify or change job card, sysout classes and trace levels. Enter NO if you do not want to be prompted.

Steplib and Utility
Enter YES if you want to be prompted to verify or change steplib and copy utility. Enter NO if you do not want to be prompted.

Procedure Names
Enter YES if you want to be prompted to verify or change the procedure names. Enter NO if you do not want to be prompted.

AD/Cycle LE/370
Enter YES if you want to be prompted to verify or change LE/370 run time options. Enter NO if you do not want to be prompted.

Enter the appropriate values and press Enter to commit the changes and return to panel EKYCP20E. Press the End key to undo changes, or the Help key to request more information.

Panels for the CHECK control statement

After you specify or verify the CCU defaults in the SETUP panels, you can start generating JCL for the CCU.

The main selection panel EKYCP00E lets you choose to generate JCL in one of two ways:

blank (GENERATE)

Use panel EKYCP10E, described on page 549, to generate and edit JCL.

S (GENSAVE)

Use panel EKYCP05E, shown in Figure 413 to store generated JCL in a library of your choice, and edit it at any subsequent time.

Panel EKYCP05E—Store JCL

If you choose blank (GENERATE) from panel EKYCP00E, you go to panel EKYCP05E, shown in Figure 413. Use this panel to specify the data set name in which to store the generated CCU job control stream. The library, or other data set, must already exist. You can store the job control stream either in an ISPF library, or another partitioned or sequential data set.

```
EKYCP05E ------------------ Save JCL in Library ------------------------------
Command ===>


Specify the library to contain the generated JCL.

   ISPF Library:
      Project ===>
      Group   ===>
      Type    ===>
      Member  ===>

   Other Partitioned or Sequential Data Set:
      DS Name ===>

Do you want to overwrite an existing member or sequential data set?

   Overwrite  ===>            (Yes or No)



Press: End to exit         Help for more information         Enter to continue
```

Figure 413. Panel EKYCP05E: Specify the JCL Library

The parameters in panel EKYCP05E are:

ISPF Library

Enter the project, group, and type of the library, and a member name under which the job control should be stored. If the member name already exists in the named library, the value you specify for the Overwrite parameter defines whether or not to replace the existing member.

Other Partitioned or Sequential Data Set

Enter the name of another partitioned or sequential data set. You can enter any fully qualified data set name by enclosing it in apostrophes. If you omit the apostrophes, your TSO prefix is left-appended to the data set name.

If you enter the name of a partitioned data set, also enter a member name under which the job control should be stored. If the member name already exists in the named partitioned data set, the value you specify on the Overwrite parameter defines whether or not to replace the existing member.

If you enter the name of a sequential data set, ensure that the data set has the following attributes:

Record format:
`FB`

Record length:
`80`

Block size: `any multiple of record length`

Overwrite
Use this field to specify whether or not you want the data set containing the generated job control stream to replace an existing data set name.

Enter YES if you want to overwrite the existing member or data set.

Enter NO if you do not want to overwrite the existing member or data set. You must then define a unique name for the new data set.

Enter the appropriate values and press either:
- ENTER to commit the changes and continue on panel EKYCP10E
- END to undo the changes and return to panel EKYCP00E
- Help to request more information

Panel EKYCP10E: CHECK control statement Parameters Part 1

You can enter the CHECK control statement with panels EKYCP10E, EKYCP11E, and EKYCP12E, shown in figures 414. through 418. On panel EKYCP10E you use the input lines to specify values for the DBD=, SEG=, EXCLUDE=, ASSIGN=, PRSET=, TAB=, PR=, and USE= operands of the CHECK control statement. See “The Check control statement ” on page 460 for a description of the CHECK control statement operands, or press the Help key.

```
EKYCP10E ----------- Enter CCU Check Command (1 of 2) ------------------------
Command ===>

Specify DBD, SEG, EXCLUDE, ASSIGN, PRSET, TAB, PR and USE operands below:
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <
 >                                                                        <

Press Enter to specify QUALIFIER, DIRECT, FORCE, READONLY, HASHONLY,
KEYONLY, MAXERROR, PSBNAME, PROCOPT, KEYLEN on the next panel.
```

Figure 414. Panel EKYCP10E: CCU Check Command (1 of 2)

An example of a CHECK input statement (part 1) is:

```
CHECK DBD=CUSTOMER, SEG=(CUST,PARTNER,VTX),
    ASSIGN=(CUST=PRCUST),
    USE=(PROD.CUST=PROD_VIEW.CUST)
```

Figure 415. Check Input Statement (Part 1)

The CCU front end does not validate the CHECK statement. The CCU validates it during the CCU initialization phase.

If you enter a question mark (?) as the value for the SEG, EXCLUDE, ASSIGN, TAB, or PR operands (for example, SEG=?), the CCU gets the information from the IMS DPROP directory, formats it in an ISPF temporary table, and then displays it in panel EKYCP12E. Then you can select from a list of choices for that operand, such as segment names to use in the SEG= list). To do so, DB2 must be started and the IMS DPROP directory for the relevant IMS DPROP system must be available to the CCU front end. You also need the authority to:

- SELECT from the IMS DPROP system directory tables
- EXECUTE Plan EKYN010X

Enter the appropriate values and press either:

- ENTER to commit the changes and continue on panel EKYCP11E
- END to undo the changes and return to panel EKYCP00E
- Help to request more information

Panel EKYCP11E—CHECK control statement Parameters Part 2

The second CHECK input command entry panel is:

```
EKYCP11E ----------- Enter CCU Check Command (2 of 2) ------------------------
Command ===>

Select options by entering either YES or NO.
   DIRECT Technique    ===>
      READONLY Option  ===>              ( Direct  technique only )
      FORCE Option     ===>              ( Direct  technique only )
   HASHONLY Option     ===>              ( Hashing technique only )
   KEYONLY Option      ===>

Enter/Verify the following parameters:
   QUALIFIER           ===>
   MAXERROR            ===>              ( 1 to 9999 )
   PSB Name            ===>
       PCB PROCOPT     ===>
       PCB KEYLEN      ===>


_____________________________________________________________________________

IMS Environment Region ===>              ( Batch or BMP )

Press: End to exit         Help for more information         Enter to continue
```

Figure 416. Panel EKYCP11E: CCU Check Command (2 of 2)

See “The Check control statement ” on page 460 for a description of these operands, or press the Help key to request tutorial pages describing these operands.

The parameters on panel EKYCP11E are:

Direct Technique

Enter YES to add the DIRECT operand to the CHECK input command and generate job control for the direct technique. `synchronous`

Enter NO for job control using the hashing technique.

READONLY Option

Enter YES to add the READONLY operand to the CHECK input command. Both the IMS and DB2 data you are going to check must be in a read-only status. `synchronous`

Enter YES only when you set the DIRECT option to YES, otherwise, enter NO.

FORCE Option

Enter YES to add the FORCE operand to the CHECK input command. `synchronous`

Enter YES only when you set the DIRECT option to YES, otherwise, enter NO.

HASHONLY Option

Enter YES to add the HASHONLY operand to the CHECK input command. `asynchronous`

Enter YES only when you set the DIRECT option to NO, otherwise, enter NO.

KEYONLY Option

Enter YES to add the KEYONLY operand to the CHECK input command; otherwise, enter NO.

QUALIFIER

Optional. Specify a value to use as the QUALIFIER= operand of the CHECK input statement.

MAXERROR

Enter a numeric value between 1 and 9999 for the MAXERROR operand, where 9999 is *all* errors.

PSB Name

By default, the CCU displays a PSB name as it was defined on setup panel EKYCP2IE (described on page 533). You can override this PSB name with an alternative name of up to 8 alphanumeric characters. The first character must be alphabetic. The value you enter is used for the PSBNAME= operand in the CHECK input command.

PCB PROCOPT

By default, you see a PCB processing option as it was defined on Setup Panel EKYCP2IE (described on page 533). For the current JCL generation, you can overwrite the default processing option with an alternative processing option. If you do, enter any valid combination of the values G, E, S, O, (T) and H (see IMS/ESA Utilities Reference: System for more information about these values). The value you enter is used for the PROCOPT= operand in the CHECK input command.

PCB KEYLEN

By default, the CCU displays a PCB key length as it was defined on Setup Panel EKYCP2IE (described on page 533). You can override the default PCB key length with an alternative key length. If you want to use the KEYLEN value determined by the CCU, leave the field blank. If you enter a value, use a number between 1 and 99999.

IMS Environment Region
Enter BATCH to generate JCL an IMS batch region. Enter BMP to generate JCL in an IMS BMP region.

Enter the appropriate values and press either:
- ENTER to commit the changes
- END to undo the changes and return to panel EKYCP00E
- HELP to request more information

After you press Enter, the panel you see next depends on the IMS DPROP system and IMS region for which you are generating JCL:

Synchronous IMS DPROP system

IMS Batch Region:
- If you set the IMS database allocation in panel EKYCP2IE (see page 533) to JCL, you see panel EKYCP30E (described on page 554), where you can specify the IMS database DD statements.
- If you set the IMS database Allocation in panel EKYCP2IE to DYN, you see the generated job control in ISPF edit mode, where you can submit it.
- You see the generated job control in ISPF edit mode, where you can submit it.

Asynchronous IMS DPROP system
- If you set the IMS database allocation in panel EKYCP21E (see page 533) to JCL, you go to panel EKYCP30E (described on page 554), where you can specify the IMS database DD statements, or the name of an HD unload file.
- If you set the IMS database allocation in panel EKYCP2IE to DYN, you go to panel EKYCP31E, (described on page 555), where you can specify the name of an HD unload file.

Panel EKYCP12E—Example of SEG Operand List

The following CHECK input statement produces panel EKYCP12E, shown in Figure 418 on page 553.

```
CHECK DBD=CUSTOMER, SEG= ?,
    ASSIGN=(CUST=PRCUST),
    USE=(PROD.CUST=PROD_VIEW.CUST)
```

Figure 417. Check Input Statement: SEG Operand List

```
EKYCP12E ------------- Select Item for Operand SEG= ---------- ROW 1 TO 4 OF 4
Command ===>                                                  SCROLL ===>

Select items for operand SEG=

Command Line:                              Column S below:
   S - selects all items.                     S - selects an item.
   D - deselects all selected items.          D - deselects a selected item.

Press the End key when finished.
                                                                    MAP-  WHERE-
S DATABASE PRSET    SEGMENT  PRID     DB2 TABLE NAME                CASE  OPTION
_ CUSTOMER PRSET1   CUST     PRCUST   PROD.CUST                    1
_ CUSTOMER PRSET1   PARTNER  PRPART   PROD.PARTNER                 2
_ CUSTOMER PRSET1   PARTSPEC PRPART   PROD.PARTNER                 2
_ CUSTOMER PRSET1   VTX      PRVTX    PROD.VTX                     3     YES
****************************** BOTTOM OF DATA ******************************
```

Figure 418. Panel EKYCP12E: Select Item for the SEG= Operand

Panel EKYCP12E lists items that are available when you specified SEG=? in the CHECK input command. You can select and deselect items either:

From the command line:

S Selects all items

D Deselects all selected items

From the S column:

S Selects one specific item

D Deselects a selected item.

Selected items replace the question mark (?) of the operand. For example, SEG=? becomes SEG=(CUST,PARTNER) if you select the segments CUST and PARTNER in the item list.

This process is repeated as long as the procedure detects a question mark (?) as a value in the CHECK input statement. Enter the appropriate values and press either:

- ENTER, after you resolve all question marks, to continue processing on panel EKYCP11E.
- END to undo the changes and return to panel EKYCP10E. You must first deselect all selected items.
- HELP to request more information.

IMS Database Allocation

On Setup Panel EKYCP2IE (see page 533) you can allocate the IMS database either dynamically or in the JCL by specifying JCL or DYN in the database allocation parameter.

If you are generating JCL for a IMS DPROP SYSTEM and you set the IMS Database Allocation parameter in panel EKYCP2IE (see Figure 402 on page 533) to JCL, the CCU displays panel EKYCP30E, shown in Figure 419 on page 554.

If you are generating a job control stream for a synchronous IMS DPROP system to submit in an IMS batch region, and you do not use the dynamic IMS database allocation, you go to panel EKYCP30E, shown in Figure 419.

```
EKYCP30E --------- Enter IMS Data Allocation Statements -------------   ---------
Command ===>
Enter IMS Database Allocation Statements:
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>



 Press: End to exit      Help for more information      Enter to continue
```

Figure 419. Panel EKYCP30E: IMS Data Allocation Statements Synchronous

Enter the IMS database allocation DD statements for the database you are going to check. For every IMS database data part, enter the DD name and the data set name. If the IMS database is an index database (HIDAM, HISAM, and so on), enter the index DD name and the name of the primary index data set. You can make unused IMS database DD statements inactive by specifying the "//*" characters in the first 3 bytes of every input line.

Enter the appropriate values and press Enter to accept values and continue. Press the End key to return to panel EKYCP00E, or press the Help key to request more information.

If you press Enter, you see the generated job control in ISPF edit mode, where you can submit it.

If you are generating JCL for an asynchronous IMS DPROP system, and you set the IMS database allocation parameter in panel EKYCP2IE to JCL you go to panel EKYCP30E, shown in Figure 420 on page 555.

```
EKYCP30E --------- Enter IMS Data Allocation Statements ----------------------
Command ===>

Enter IMS Database Allocation Statements:
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>
   ===>

Processing an Asynchronous DPROP System:
   You may enter the name of an HD Unload File to be used as a
   replacement for the IMS database.

   HD Unload File DSN ===>


Press: End to exit         Help for more information         Enter to continue
```

Figure 420. Panel EKYCP30E: IMS Data Allocation Statements Asynchronous

Use this panel to enter the IMS database allocation DD statements for the database you are going to check, or enter the name of an HD unload file to be used as a replacement for the IMS database. The CCU changes all IMS database DD statements defined in a previous call of the CCU front end to JCL comment lines by adding "//*" to the first 3 bytes of every input line.

If you are not using an HD unload file:

For every IMS database data part, enter the DD name and the data set name. If the IMS database is an index database, such as HIDAM or HISAM, enter the index DD name and the primary index data set name. You can make unused IMS database DD statements inactivate by specifying the "//*" characters in the first 3 bytes of every input line.

If you are using an HD unload file:

Enter the name of the HD unload file to use in the IMS read phase to retrieve the IMS segments. Change all IMS database DD statements to JCL comment lines by adding "//*" characters to beginning of every input line.

Enter the appropriate values and press either:
- ENTER to commit the changes and continue on panel EKYCP31E
- END to undo the changes and return to panel EKYCP00E
- Help to request more information

Panel EKYCP31E—HD Unload File Names

If you are generating job control for an asynchronous IMS DPROP system, and you set the IMS database allocation parameter in panel EKYCP2IE to DYN, the CCU displays panel EKYCP31E, shown in Figure 421 on page 556.

```
EKYCP31E -------------- Enter HD Unload File Name ----------------------------
Command ===>

   Processing an Asynchronous DPROP System

      You may enter the name of an HD Unload File to be used as a
      replacement for the IMS database.

      HD Unload File DSN ===>









Press: End to exit         Help for more information         Enter to continue
```

Figure 421. Panel EKYCP31E: HD Unload File Names

Enter the name of the HD unload file to use in the IMS read phase to retrieve the IMS segments. You can submit the JCL from this screen.

Enter the appropriate values and press either:
- ENTER to commit the changes continue on panel EKYCP40E
- END to undo the changes and return to panel EKYCP00E
- Help to request more information

Panel EKYCP40E—Query IMS DPROP Directory

You query the IMS DPROP directory to retrieve information to use to create a CHECK control statement for the CCU.

To execute the query function you need:
- The SELECT privilege for the IMS DPROP directory tables DPRPR, DPRSEG, DPRTAB and DPRFLD
- Authority to EXECUTE DB2 plan EKYN020X

Option Q (QUERY) in the main selection panel, EKYC000X, (see page 529), takes you to panel EKYCP40E, shown in Figure 422 on page 557. Use this panel to specify the type of information you want to retrieve.

```
EKYCP40E ------------------ Query DPROP Directory ----------------------------
Command ===>

Select the type of information you want to retrieve:
  1 - Display General Overview
  2 - Display whether or not direct technique is supported
  3 - Display Field Definitions

For any selection above, retrieve DPROP information relating to:
   DPROP Directory Qualifier   ===>           (Qualifier, or ? for list)

   Enter
      either PRID              ===>

      or     IMS Database Name ===>          and/or Segment ===>

      or     DB2 Table Prefix  ===>
             DB2 Table Name    ===>

Output Destination             ===>           (BROWSE or sysout class)

Press: End to exit         Help for more information         Enter to continue
```

Figure 422. Panel EKYCP40E: Query IMS DPROP Directory

Select the type of information you want to retrieve
Specify 1, 2, or 3 in the command line:

1 - Display General Overview
Displays a list of the PRIDs, segments, and DB2 tables that are available in the IMS DPROP directory.

2 - Display whether or not direct technique is supported
Displays a list of the PRIDs that are supported in the direct technique, and those that are not. However, the information within the IMS DPROP directory is not complete, and you need to know the IMS database characteristics to determine whether or not the direct technique is supported.

3 - Display Field Definitions
Displays a list of the PRIDs, segments, DB2 tables and their IMS fields, and DB2 columns that are available in the IMS DPROP directory.

See the Query Report panel EKYCP41E on page 559 for examples.

IMS DPROP Directory Qualifier
Specify the name of the IMS DPROP directory table from which you want to get the information. To receive a list of DB2 qualifiers that are available in the current DB2 system, enter a question mark (?) and press Enter.

PRID
Enter the PRID you want to display. To receive information for all PRIDs, enter a “%” character only.

IMS Database Name or Segment Name
Enter the IMS database or segment name. To receive information for all databases and segments, enter a “%” character only.

DB2 Table Prefix and/or Table Name
Enter the DB2 table prefix or the DB2 table name or both. To receive information for all DB2 tables enter a “%” character only.

You must supply either the PRID, or IMS database name, or the DB2 table prefix or name.

Output Destination

To receive the report on your terminal, enter BROWSE.

If you want to route the report to the printer, enter any valid output class.

Enter the appropriate values and press either:

- ENTER to commit the changes and process the query request.
- END to return to panel EKYCP00E
- Help to request more information

Panel EKYCP41E—QUERY Report Option 1

Option **1** in panel EKYCP40E results in the report shown on panel EKYCP41E in Figure 423.

```
EKYCP41E ----------------------- Query Report ------ LINE 00000000 COL 001 080
COMMAND ===>                                                  SCROLL ===> PAGE
SEARCH FOR: PRID Name like '%'

        Mapping  Mapping  Segment   Segment      Table     Table
 PRID    Case     Dir     Name      Role         Qual      Name
****************************** TOP OF DATA  ******************************

DBD Name = CUSTOMER , PRSET Name = PRSET1


 PRCUST     1      TW     CUST      ENTITY       PROD      CUST

 PRPART     2      TW     CUST      PARENT       PROD      PARTNER
                          PARTNER   ENTITY
                          PARTSPEC  EXTENSION

 PRVTX      3      TW     CUST      PARENT       PROD      VTX
                          PARTNER   PARENT
                          VTX       ENTITY

****************************** BOTTOM OF DATA ******************************
```

Figure 423. Panel EKYCP41E: Query Report - Option 1

Press the End key to return to panel EKYCP40E.

Panel EKYCP41E—QUERY Report Option 2

Option **2** in panel EKYCP40E produces the report shown in Figure 424 on page 559.

```
EKYCP41E ----------------------- Query Report ------ LINE 00000000 COL 001 080
COMMAND ===>                                                   SCROLL ===> PAGE
SEARCH FOR: PRID Name like '%'

   DIRECT            Segment   Segment     Table     Table
 Supported  PRID     Name      Role        Qualifier Name
******************************** TOP OF DATA  *********************************

DBD Name = CUSTOMER , PRSET Name = PRSET1

    YES     PRCUST   CUST      ENTITY      PROD      CUST
    YES     PRPART   PARTNER   ENTITY      PROD      PARTNER
    YES     PRPART   PARTSPEC  EXTENSION   PROD      PARTNER
    NO      PRVTX    VTX       ENTITY      PROD      VTX

****************************** BOTTOM OF DATA ******************************
```

Figure 424. Panel EKYCP41E: Query Report - Option 2

Press the End key to return to panel EKYCP40E.

Panel EKYCP41E—QUERY Report Option 3

If you select option **3** in panel EKYCP40E, and enter PRID name “PRPART,” you see panel in Figure 425.

```
EKYCP41E ------------------------------- Query Report ----------------- LINE 00000000 COL 001 080
COMMAND ===>                                                                  SCROLL ===> PAGE
SEARCH FOR: PRID Name = 'PRPART'

    Segment            Field    Field     ID-   Field                  Primary
    Name     Role      Name(8)  Type     Field? prop? DB2 Column Name  Key?    Nulls     Ordering
*********************************************************************************************

DBD Name = CUSTOMER , PRSET Name = PRSET1

 PRID = PRPART , Table Name = PROD.PARTNER

    CUST     Parent     FCUNBR   Key       No    Yes   CCUNBR           Yes     No        Asc
    PARTNER  Entity     FPATNR1  Key       No    Yes   CPATNR1          Yes     No        Asc
                                 Data      No    No    CPATNR2          No      With Def  -
                        FCONTEXT Data      No    Yes   CCONTEXT         No      With Def  -
                                 Data      No    No    CPATNR3          No      With Def  -
    PARTSPEC Extension  FCONTEXT Data      No    Yes   CCONTEXTX1       No      With Def  -

***************************************** BOTTOM OF DATA ****************************************
```

Figure 425. Panel EKYCP41E - Query Report - Option 3

Press the End key to return to panel EKYCP40E.

Part 8. Audit Extract Utility

Chapter 21. Audit Extract Utility

This topic describes the Audit Extract utility restrictions, input and output, environment, and how to call the utility.

Use the Audit Extract utility (AEU) to extract audit trail records from a sequentialinput file and load them into the audit trail table. You can then query the audit trail table to obtain information on data propagation events.

The following topics provide additional information:
- “Input and Output ”
- “Environment ”
- “JCL Requirements ”
- “Output Messages and Statistics ” on page 565
- “Return Codes and Error Conditions ” on page 565

Input and Output

You can load the IMS DPROP SMF records into the audit trail table in two ways:
- Load the SMF records into the table every time the SMF data sets are dumped using IFASMFDP.
- Load the SMF records on an ad hoc basis, such as whenever a problem needs to be investigated.

Environment

The AEU is a DB2 application program that uses the TSO attach facility to connect to DB2. You invoke it with the RUN PROGRAM subcommand of the DSN command processor. If you have a Selector-only site where DB2 is unavailable, you cannot run the AEU. You can extract the appropriate SMF records to a sequential file using IFASMFDP, transfer this file to the Receiver site and run the AEU there, but it is your responsibility to provide a transfer program.

The AEU only inserts rows in the audit trail table. For maintenance purposes, you must provide your own SQL application or use QMF to remove old data from the table.

The AEU sequential input file must be created by the SMF dump program IFASMFDP. For more information on this program, see*OS/390 MVS System Management Facilities*.

JCL Requirements

Calling the AEU is a two-step job:
- The first step is to call IFASMFDP to dump the SMF records from the VSAM data sets to the sequential output files.
- The second step is to invoke AEU to insert the sequential output files into the audit trail table.

Figure 426 on page 564 shows the JCL for this two-step process.

```
//STEP1     EXEC  PGM=IFASMFDP
//INDD1     DD    DSN=SYS1.MANx,DISP=SHR
//OUTDD1    DD    DSN=SMFDATA,UNIT=TAPE,DISP=SHR,VOL=SER=SMFTAP
//OUTDD2    DD    DSN=&&TEMP,DISP=(,PASS),UNIT=SYSDA,
//     SPACE=(CYL,(5,10))
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
   INDD(INDD1,OPTIONS(ALL))
   OUTDD(OUTDD1,TYPE(000-255))
   OUTDD(OUTDD2,TYPE(nnn,00,02,03,07,90))
//*
//STEP2     EXEC  PGM=IKJEFT01
//EKYRESLB  DD    DSN=DPROP.LOAD,DISP=SHR
//SYSTSPRT  DD    SYSOUT=*
//SYSTSIN   DD    *
    DSN SYSTEM(DSN)
    RUN PROGRAM(EKYAUD00) PLAN(planname) LIB('DPROP.LOAD')
//AEUT1    DD    DSN=&&TEMP,DISP=(OLD,PASS)
//AUDPRINT  DD    SYSOUT=*
```

Figure 426. Audit Extract Utility: Sample JCL

In step 1, the first SYSIN OUTDD control statement specifies that all SMF records should be written to the file described by the //OUTDD1 DD statement. This file is usually available to dump SMF records.

Although any SMF record type can be passed as input to the AEU, only the AEU uses the IMS DPROP record type and the system record types 00, 02, 03, 07, and 90. Therefore, a file containing only records of these types is created for AEU input. The second SYSIN OUTDD control statement specifies that the record types appropriate to the AEU be written to the sequential data set specified in the //OUTDD2 DD statement. These are system record types 00, 02, 03, 07, 90, and the particular SMF record type indicated by the value *nnn* in Figure 426 assigned for IMS DPROP use during the preparation of MVS. For more information on the IMS DPROP SMF record type see *IMS DPROP Installation Guide*.

When DB2 is unavailable in a Selector-only site, the sequential file specified on the //OUTDD2 DD statement should be a permanent file which can then be transferred to the Receiver site for AEU input.

For more information on the JCL for step 1, see*OS/390 MVS System Management Facilities*.

In step 2, the JCL invokes AEU as a DB2 TSO batch application. The DD statements used in step 2 are described below. The //EKYRESLB DD statement is common to many IMS DPROP components and is described in detail in Chapter 1, “Common JCL for IMS DPROP Components,” on page 3.

//SYSTSPRT DD Statement
A print file used by TSO batch.

//SYSTSIN DD Statement
Used as input by TSO batch.

The standard DB2 DSN command invokes the DSN command processor. The SYSTEM keyword names the DB2 subsystem. You can use standard DB2 keywords such as RETRY or TEST on the DSN command.

The DB2 RUN subcommand identifies:

- The name of the DB2 application to be called. The PROGRAM keyword contains this name; and for the AEU it is EKYAUD00.
- The name of the DB2 plan used for the execution of EKYAUD00. Assign this name during IMS DPROP generation.
- The load module library containing EKYAUD00, located in the LIB keyword.

//AUDPRINT DD Statement
A print file used by AEU to write error messages and other information.

//AEUT1 DD Statement
The input file containing the IMS DPROP SMF records. This file is created by the SMF dump program (IFASMFDP).

Output Messages and Statistics

All messages from the Audit Extract utility begin with EKYA. For more information on messages, see*IMS DPROP Messages and Codes*.

Return Codes and Error Conditions

The Audit Extract utility provides three return codes.

00 Normal return

04 Warning issued

08 Errors found

The audit trail table might contain incomplete audit trail data if:

- The AEU encounters an SQL error, writes diagnosis information to //AUDPRINT, and ends.
- The contents of some SMF files were not input to AEU. SMF stores data in multiple data sets that are dynamically switched. See *OS/390 MVS System Management Facilities* for information on extracting SMF data.

Errors can occur while IMS DPROP writes the audit trail records to SMF. Some errors that might occur are:

- SMF is not active.
- The record is lost due to a buffer shortage.
- The record is lost because SMF data sets are full.
- The record is not written to SMF because the SMFPRM*xx* member of SYS1.PARMLIB has not been updated to request recording of the IMS DPROP record type.
- The record has been dropped by the IEFU83 user exit routine.

Part 9. IMS-to-IMS Replication

Chapter 22. IMS-to-IMS replication

IMS-to-IMS replication builds upon the existing MQSeries-based asynchronous feature by making an IMS Apply program available. By using the IMS Apply program and the existing IMS Capture program, IMS-to-IMS replication can be achieved.

If you are already using the MQSeries-based asynchronous IMS DataPropagator features for IMS-to-DB2 propagation, you are familiar with Capture processing. Consequently, you need to become familiar only with the IMS Apply program, which shares much in common with the existing Apply program supports IMS-to-DB2 propagation. If you are not already familiar with the existing MQSeries-based features of IMS DataPropagator, you will need to understand how to use the existing Capture program, which is documented in Chapter 14, “Capture Component,” on page 363.

The following topics provide additional information:
- “IMS-to-IMS replication Capture component”
- “The IMS Apply program”

IMS-to-IMS replication Capture component

The IMS Capture program is the same for both IMS-to-DB2 propagation and for IMS-to-IMS propagation. Normally, the IMS Capture program cannot detect if it is capturing data for subsequent DB2 or for subsequent IMS propagation. Therefore, you can use the IMS Capture program to propagate to either one or both DB2 and IMS targets.

Refer to Chapter 14, “Capture Component,” on page 363 for more information about the Capture component and its setup.

The IMS Apply program

The IMS Apply program uses both MQSeries and IMS services. The IMS Apply program is an IMS BMP that does the following functions:
- Uses MQSeries to read the MQSeries messages that are sent by the IMS Capture component
- Updates the target IMS databases with captured changed IMS source data that is found within the MQSeries messages

The IMS Apply program occurrences are started like other jobs or started tasks in your installation (often with standard automated operation tools of your installation). IMS Apply program occurrences are stopped in one of the following two ways:
- By issuing a `STOP` *`jobname`* command
- When the MQSeries Queue Manager stops

IMS Apply program occurrences are permanently active when they are used for near real-time propagation. When IMS Apply program occurrences are used for point-in-time propagation, they are started by the system operator when the target IMS databases must be updated.

You can run either one single occurrence or multiple occurrences of the Apply program. Running multiple Apply program occurrences can increase throughput. To

run multiple Apply occurrences concurrently, you can split the transmission of the changed source IMS data into multiple PRSTREAMs by using KEYRANGE= control statements.

Note: A particular MQSeries input queue and a particular PRSTREAM cannot be concurrently processed by more than on Apply program occurrence. This preserves the FIFO processing sequence of the MQSeries messages that contains the changed IMS source data.

An Apply program occurrence uses a control statement file to understand which MQSeries queue is used as input. This MQSeries queue is opened for exclusive use by the current Apply program occurrence.

Processing MQSeries messages that contain IMS changes

When the IMS Apply program receives an MQSeries message that contains an IMS change, it calls an IMS update program to update the target IMS databases.

IMS Apply program JCL

The IMS Apply program runs as an IMS BMP program. Use the JCL procedures that are provided by IMS for IMS BMP job steps. Figure 427 is an example of the IMS Apply program JCL.

```
//IMSAPPLY EXEC IMSBATCH,
//     MBR=EKYMQIAP,
//     PSB=MYAPPLY,
//     IMSID=MYIMSID,
//     AGN=MYAGN
//STEPLIB   DD     ......................
//          DD     ......................
//DFSRESLB  DD     ......................
//IMS       DD     ......................
//          DD     ......................
//PROCLIB   DD     ......................
//EKYRESLB  DD     DSN=PREFIX.EKYRESLB,DISP=SHR
//DFSESL    DD     ......................
//          DD     ......................
//APPLYN   DD     *
APPLY       NAME=APPLY1;
QMANAGER    NAME=MQD1;
QUEUE       NAME=MQ.APPLY.QUEUE01;
//SYSPRINT  DD     SYSOUT=*
//SYSUDUMP  DD     SYSOUT=*
//EKYIN     DD     DUMMY
//EKYPRINT  DD     SYSOUT=*
//EKYTRACE  DD     SYSOUT=*
```

Figure 427. IMS Apply program JCL example

//STEPLIB
If the IMS DPROP modules, IMS modules, or MQSeries modules are not in the link pack area or in a link-list library, you must provide a //STEPLIB or //JOBLIB to identify the library in which they are located.

//EKYRESLB
This DD statement is common to many IMS DPROP components. Refer to "//EKYRESLB DD Statement" on page 4 for more information.

//DFSESL
This DD statement might be required by IMS. Refer to "//DFSESL DD Statement " on page 4 for more information.

//APPLYIN
This DD statement defines the mandatory file where you must provide control statements for the Apply program.

//EKYIN
This DD statement defines an optional file where you can provide TRACE and TRDEST control statements. This statement is used to activate the IMS DPROP trace, usually for problem determination purposes.

//EKYPRINT
This DD statement defines a SYSOUT data set containing messages from the Apply program (and from other IMS DPROP components).

//EKYTRACE
This DD statement defines the usual trace data set of IMS DPROP. Refer to "//EKYTRACE DD Statement " on page 7 for more information.

//APPLYIN DD statement

This statement is mandatory. It contains the most important definitions of the IMS Apply program. Refer to "Control statements in the //APPLYIN file" on page 572 for more information about the control statements of the //APPLYIN file.

Each occurrence of the Apply program must use its own //APPLYIN file. The //APPLYIN file must be a sequential file or a PDS Member and must have the following characteristics: RECFM=F or FB, and LRECL=80.

The //APPLYIN file contains the following mandatory control statements:

- APPLY control statement
- QMANAGER control statement
- QUEUE control statement

The //APPLYIN file contains the following Optional control statements:

- 1 to *n* FAILURES control statements
- 1 to *n* NONUNIQUE control statements
- 1 to *n* optional VERIFY control statements
- 1 to *n* optional STOPAT control statements
- 1 optional STATISTIC control statement
- 1 optional COMMIT control statement

//APPLYIN file examples

Example 1 – //APPLYIN file supplied as a PDS member: The example in Figure 428 shows the //APPLYIN file specified as PDS data set member APPLY01.

```
//APPLYIN DD DSN=IMSDPROP.APPLYDEF(APPLY01),DISP=SHR
```

Figure 428. Apply program //APPLYIN DD statement example 1

Example 2 – //APPLYIN example showing mandatory control statements: The example in Figure 429 shows an //APPLYIN file example with the mandatory control statements APPLY, QMANAGER, and QUEUE specified.

```
 //APPLYIN DD *
APPLY NAME=IMSAPPLY;
QMANAGER NAME=MQD1;
QUEUE NAME=MQ.APPLY.QUEUE;
/*
```

Figure 429. Apply program //APPLYIN DD statement example 2

Example 3 – //APPLYIN example showing mandatory and optional control statements: The example in Figure 430 shows an //APPLYIN file example with the mandatory control statements APPLY, QMANAGER, and QUEUE and the optional control statements VERIFY and FAILURES.

```
 //APPLYIN DD *
APPLY NAME=IMSAPPLY;
QMANAGER NAME=MQD1;
QUEUE NAME=MQ.APPLY.QUEUE;
VERIFY DBD=PARTSDB;
FAILURES CATEGORY=DATA,ACCEPT=100;
/*
```

Figure 430. Apply program //APPLYIN DD statement example 3

//EKYIN DD statement

The optional //EKYIN statement provides IMS DPROP tracing-related specifications. For the Apply program, the following control statements can be provided in the //EKYIN file:

- TRACE
- TRDEST

Other JCL statements for the IMS Apply program

Other JCL statements for the IMS Apply program include:

- //JOBLIB or //STEPLIB
- //EKYRESLB
- //EKYPRINT

Control statements in the //APPLYIN file

APPLY control statement

Only one APPLY control statement can be provided in the //APPLYIN file.

The APPLY control statement assigns a unique name to each occurrence of the Apply program. When the APPLY program starts, it verifies that another occurrence of the Apply program with the same name is not currently executing.

For syntax of the APPLY control statement, refer to“APPLY control statement” on page 379.

NAME= *name_of_apply_program*
Assigns a unique name to the Apply program occurrence. The name must consist of 1-8 alphanumeric or the $, #, @ special characters. The first character cannot be numeric.

COMMIT control statements

The optional COMMIT control statement overrides the default commit frequency of the Apply program. If you provide multiple COMMIT control statements, specifications on the *n*th control statement override specifications on the *m*th control statement (n>m).

For the syntax of the COMMIT control statement refer to “COMMIT control statements” on page 379.

AFTER= *nnnnn*
Specifies that the Apply program commits, approximately, after processing *nnnnn* database updates.

The default value is 500. The maximum value is 999999.

FAILURES control statements

As with other functions, IMS DPROP distinguishes between different types of failure categories. This capability allows you to accept certain types of data-related failures while avoiding the risk of losing or discarding changed data if a target resource is accidentally temporarily unavailable. You can limit the maximum number of Failure occurrences that IMS DPROP will accept before the program ends. This capability is useful because it might be OK to accept a couple of failures, but unacceptable to continue the processing if thousands of database records are affected. It is also possible to limit the number of Failure occurrences that IMS DPROP will report. Limiting failure occurrences can be used to avoid to flood Consoles, Print Files, and the Error DB with thousands of messages and records.

Use the optional FAILURES control statements to override default options or settings of the error handling logic of the Apply program. They can be used to:
- Tell the APPLY program to accept a limited number of errors
- Tell the IMS Apply program to limit the number of propagation failures that will be documented with error messages, inserts into the ERROR Table, and traces

Error Categories Recognized by IMS Apply: The IMS Apply program distinguishes between the following category of errors:
- Data Errors
- Unavailability problems
- Miscellaneous Errors
- Should-Not-Occur Errors

For example a data error might be encountered when the IMS Apply program is processing the Replace of an IMS segment occurrence and attempts to replace the corresponding segment in the target database does not exist.

Default Error Options: If no FAILURES control statements are provided, the APPLY program will abend.

Overriding the Default Error options: The default actions can be overridden by providing a FAILURES statement.

Data Errors: By using the optional VERIFY control statement of the IMS Apply program, an error condition might be flagged if the segment data of a target segment to be replaced or deleted does not match the segment data within the MQSeries update message.

Overriding the Default Error Options for data errors: The default action can be overridden by providing a FAILURES CATEGORY=DATA, ACCEPT=*nnnn*,REPORT=*nnnn* control statement.

Unavailability Problems: When the IMS Apply program encounters an unavailability problem, it writes error messages and saves the message in the EKYERREB database. The IMS Apply program then reads the next message. If the number of data errors exceeds the value specified, the Apply program abends.

Overriding the Default Error Options for Unavailability errors: The default action can be overridden by providing a FAILURES CATEGORY=UNAVAILABLE, ACCEPT=*nnnn*,REPORT=*nnnn* control statement.

Miscellaneous Errors: When the Apply program encounters a miscellaneous problem, it writes error messages and abends. You should investigate the reported problem, fix the problem, and restart the Apply program.

Examples of what the Apply program calls miscellaneous errors include the reading of an MQSeries message that has not been created by IMS DPROP (this occurs with MQSeries queues that are not used exclusively by IMS DPROP). It also includes readings of MQSeries messages that do not have the expected IMS DPROP message format.

Overriding the default error options: The default actions can be overridden by providing a FAILURES CATEGORY=MISC, ACCEPT=*nnnn*; control statement. The Apply program processes the control statement in the same way as for mapping and unavailability errors, described earlier in this topic.

Should-Not-Occur Errors: A should-not-occur error involves an IMS DPROP logic error. When the Apply program encounters this type of error, it writes error messages and abends.

Warning: Use of FAILURES CATEGORY=*xxxxxx*, ACCEPT=*nnnnn* control statements typically results in IMS changes that are not propagated. Instead, these IMS Changes are stored for problem determination purposes by the IMS Apply program in a special IMS database: the MQ-ASYNC ERROR database (ERRORDB). Because some changes are not propagated, the source and target copy of the propagated data become inconsistent. To reinstate the consistency, you must individually identify and perform the update to the target database or repeat an initial load of the target database.

Recommendation: Use the FAILURES CATEGORY=*xxxxxx*, ACCEPT=*nnnnn*; control statements only as a last resort or when the consistency of source and target copies is not important.

Note: When the Apply program abends because of a data, unavailability, or miscellaneous error, its abend message identifies the category of the encountered error. The reported error category can then be used to specify the CATEGORY= keyword value on a FAILURE control statement if the operator decides to restart the Apply program with a FAILURE control statement.

If you provide multiple FAILURES control statements, specifications on the *n*th control statement override specifications on the *m*th control statement (*n*>*m*).

For syntax of the FAILURES control statement, refer to “FAILURES control statements” on page 382. Note that the MAPPING category is not relevant or available for FAILURES with the IMS Apply program and is replaced by the DATA category.

CATEGORY= *xxxxx* **, ACCEPT=***nnnn*

Tells the Apply program to accept, within each 15-minute cycle, up to the specified number of errors of the error category *xxxxx*.

Values for *xxxxx* include MAPPING, UNAVAILABLE, or MISC.

Substitute *nnnn* with a number in the range between 0 and 999999999. The default value is 0.

If you need to accept errors of more then one category, which is unlikely, you can provide FAILURE control statements for each error category.

The keyword specifications that follow affect the error handling of all errors and error categories, not only the error category identified on the optional CATEGORY= keyword.

MSGWTO= *9999*

Tells the Apply program to report, within each 15-minute cycle, up to the specified number of errors with WTOs (write-to-operators) on the /390 operator consoles.

The maximum number that can be specified is 999999999. The default value is 50.

Note: IMS DPROP also reports errors on a print file, on its trace file, and on its audit trail. If encountering more errors than the MSGWTO specification, the Apply program still reports errors on its print file, trace file, and audit trail (assuming that the MSGPRINT=, TRACE=, and MSGAUD= limits have not been exceeded).

MSGPRINT= *9999*

Tells the Apply program to report, within each 15-minute cycle, up to the specified number of errors on the //EKYPRINT Print file.

The maximum number that can be specified is 999999999. The default value is 100000.

MSGAUD= *9999*

Tells the Apply program to report, within each 15-minute cycle, up to the specified number of errors (through SMF records) for the IMS DPROP Audit Trail.

The maximum number that can be specified is 999999999. The default value is 100000.

TRACE= *9999*

Tells the Apply program to document, within each 15-minute cycle, up to the specified number of errors on the IMS DPROP trace.

The maximum number that can be specified is 999999999. The default value is 100000.

INSERT= *9999*

Tells the Apply program to insert, within each 15-minute cycle, up to the specified number of IMS changes that could not be propagated into the ERROR Table.

The maximum number that can be specified is 999999999. The default value is 100000.

NONUNIQUE control statements

The NONUNIQUE control statement identifies the segment types that do not have a unique key but have multiple occurrences under their parent.

The handling of such segment types are as follows:

For REPLACES and DELETES, after the GHU *C that retrieved the segment identical to the Before-Change Image sent by Capture:

- When identical a REPL/DLET call is issued.
- Otherwise a GHN call is issued until under the same parent a segment occurrence identical to Before-Change image is retrieved. Then the REPL/DLET Call is issued.
- If under the same parent a segment occurrence identical to Before-Change image is not found: this is an error situation and will be handled according to the selected “failure” processing options.

For INSERTS:

- Position by using the GHU *C call, then ISRT
- Position of inserted segment will depend on the DBDGEN FIRST/LAST/HERE insert rule. If HERE insert rule on source system, sequence of segments on Source and target will typically differ.

The following diagram shows the syntax of the NONUNIQUE control statement:

```
►►──NONUNIQUE──┬──DBD=(──┬──dbd_name──┬──)──────────────────┬──;──────────────────►◄
               │         └──────<──────┘                     │
               │                                             │
               └──,DBD=dbd_name,SEG=(──┬──seg_name──┬──)──────┘
                                       └──────<──────┘
```

DBD=*database, SEG=(segname1, segname2,...)*

Names a physical IMS segment type, or a list of IMS segment types.

NONUNIQUE control statement examples:

Example 1 – NONUNIQUE identifying two segments without unique keys within one DBD: The example in Figure 431 shows the NONUNIQUE control statement being used to identify two segments to the IMS APPLY program. The two segments are within a database and do not have unique keys, although they possibly have multiple occurrences under the same parent.

```
NONUNIQUE DBD=DB1,SEG=(SEG3, SEG5);
```

Figure 431. IMS APPLY NONUNIQUE control statement, example 1

Example 2 – NONUNIQUE identifying two DBDs each with a segment without unique keys: The example in Figure 432 shows NONUNIQUE control statements being used to identify segments to the IMS APPLY program. The segments are within two different databases each without unique keys, although they possibly each have multiple occurrences under the same parent.

```
NONUNIQUE DBD=DB3,SEG=SEG5;
NONUNIQUE DBD=DB4,SEG=SEGX;
```

Figure 432. APPLY NONUNIQUE control statement, example 2

QMANAGER control statement

This statement is mandatory. You must provide only one QMANAGER control statement in the //APPLYIN file.

The QMANAGER control statement identifies the MQSeries queue manager that is used to read the MQSeries messages that contains the changed data that was sent by the Capture component.

For QMANAGER control statement syntax, refer to “QMANAGER control statement” on page 386.

NAME= *name_of_mqseries_queue_manager*
Identifies the name of the MQSeries queue manager that will be used by the Apply Component of IMS DataPropagator to read the changed data that was sent by the Capture component.

QUEUE control statement

The mandatory QUEUE control statement identifies which MQSeries input queue contains the propagation data streams that must be processed by this occurrence of the Apply program.

Note that the queues that are used by MQ-ASYNC must be reserved for exclusive use by MQ-ASYNC. Also, in order to preserve the FIFO message processing sequence, the Apply program opens its MQSeries input queue with the INPUT_EXCLUSIVE option of MQSeries. Therefore, this queue cannot be processed concurrently by more then one occurrence of the Apply program.

Each occurrence of the Apply program can read from one and only one MQSeries input queue. One MQSeries input queue can contain one or multiple propagation data streams.

For QUEUE control statement syntax, refer to “QUEUE control statement” on page 387.

NAME= *name_of_mqseries_input_queue*
Identifies the name of the MQSeries queue (or of an MQSeries QALIAS) that will be used as input by the Apply program occurrence.

STATISTIC control statement

The optional STATISTIC control statement is used to override the default options for the periodic writing of performance statistics. If multiple STATISTIC control statements are provided, specifications on the *n*th control statement override specifications on the *m*th control statement (*n*>*m*).

For STATISTIC control statement syntax, refer to “STATISTIC control statements” on page 389.

FREQUENCY= *nnnnnn*

Specifies that after *nnnnnn* seconds the Apply program will write performance statistic records.

The default value is 3600 (an hour). The maximum value is 999999.

Note: The Apply program writes final statistics at the time of its normal completion.

DETAIL=

Specifies how detailed the performance statistics should be. DETAIL is any combination, in any order, of the following keywords:

PSB Provides detail at the PSB level

DBD Provides detail at the DBD level

SEG Provides detail at the SEG level

These selections are used to build the key for one report item on the statistics.

If the DETAIL keyword is omitted, then statistic report items are done only per the source IMS system ID.

STOPAT control statement

The optional STOPAT control statement is used to automatically stop the Apply program when it reads an MQSeries message with a source system timestamp that is higher than the specified source system timestamp.

After this automated stop of the Apply program, the data in the target IMS databases reflects the content of the source IMS databases at the specified source system timestamp. The STOPAT control statement can therefore be used to implement propagation that reflects, in the content of the target IMS databases, a clear source system point-in-time.

If you provide multiple STOPAT control statements in the //APPLYIN file, the Apply program stops when it encounters the first MQSeries message that satisfies at least one of the STOPAT control *statement.specifications*.

Guidelines for using STOPAT :

1. To reflect an accurate source system point-in-time in the target table, the source IMS databases should be quiesced at the specified point-in-time. At this point-in-time, there should be no active UOW that has updated or will update the source databases.

   If this guideline is not observed and some updating UOWs are active at the specified point-in-time, some of the updates of these UOWs might not be on the target databases after the Apply program is stopped. However, such updates are not lost; they remain in the MQSeries queue until the Apply program is started again and are processed at this later time.

   In addition, if this guideline is not followed, some IMS UOWs might start or complete either before or after the specified point-in-time; some other IMS UOWs might overspan the point-in-time. Therefore, updates of a particular UOW might or might not be included in the target tables when the Apply program stops.

2. Do not use the STOPAT control statement for an Apply program that processes multiple propagation data streams that have not been transmitted together in FIFO order from the same source system queue.

   If this guideline is not observed, when the Apply program is told to stop at 10:00, the program might read a message from stream 1 that was written at 10:01 before it reads from a message from stream 2 that was written at 9:58. It then stops when it encounters the message that was written at 10:01 before reading the message that was written at 9:58.

Note: The operators of the target system can also stop the Apply Program by issuing an OS/390 Console `STOP jobname` command.

For STOP control statement syntax, refer to “Guidelines for Using STOPAT ” on page 388. However, note that ID= shown in this diagram is not implemented for the APPLY program.

TS= *timestamp*
Tells the Apply program to stop processing when reading MQSeries messages that have been created after the specified source system timestamp.

The source system timestamp is specified based on the local time of the source system and must be provided in the following format: *yyyy-mm-dd-hh.mm.ss.nnnnnn*. The year, month, and day portions must be specified. If your timestamp omits trailing hour, minute, second, or microsecond specifications, implicit specifications of zeroes are assumed for the missing trailing portion.

The logic of the Apply is as follows: when reading an MQSeries message that has been written after the specified timestamp, the Apply program stops its processing without processing this MQSeries message. This message remains in the MQSeries queue and is processed by an eventual execution of an Apply program.

When reading an MQSeries message that has been written exactly at the specified timestamp, the Apply program processes this message and attempts to read subsequent messages. If the next MQSeries message has the same timestamp, it is processed, and the Apply program attempts to read subsequent messages. If the next message has a later timestamp, or if the Apply program does not find any further message to process, the Apply program then stops.

Note: On the source system, if there was a long period of no capture activities at and after the specified timestamp, the Apply program, when executing in near real-time mode, waits a long time before it gets an MQSeries message that has been written on the source system after the specified timestamp. If you must immediately stop the Apply program, you can use the `STOP Jobname` command.

VERIFY control statements

The VERIFY control statement requests that when propagating replaces and deletes, the IMS Apply program verifies that the data of the Before Image that was sent by the source system matches the data that is located in the segment of the target DB that will be replaced or deleted.

If unequal, the IMS Apply program considers that this is an Error of the Category Data (refer to the FAILURES control statement).

The VERIFY control statement is ignored for segments that are identified on NONUNIQUE control statements.

The following diagram shows VERIFY control statement syntax:

```
                                  ┌─,──────────┐
                                  ▼            │
►►─NONUNIQUE──┬─DBD=(──dbd_name──┴──)──────────────────┬──;──────────────────────►◄
              │                                        │
              │                                  ┌─,──────────┐
              │                                  ▼            │
              └─,DBD=dbd_name,SEG=(──seg_name──┴──)──┘
```

DBD=*database1, database2,...)*
Names a physical IMS database or a list of physical IMS databases.

DBD=*database, SEG=(segname1, segname2,...)*
Names a physical IMS segment type or a list of IMS segment types.

OS/390 Operator Commands for the IMS Apply Program

The IMS Apply program supports the following OS/390 system commands:

- STOP *jobname*
- MODIFY *jobname*,...

In these OS/390 system commands, replace *jobname* with the name of the OS/390 job of the Apply program.

STOP jobname
The P *jobname* (STOP) OS/390 system command is used to stop the Apply program.

MODIFY jobname
The F *jobname* (MODIFY) command can be used to:

- Display the current status of the Apply Program
- Display some statistics on the console
- Force a new statistic cycle
- Change the frequency interval of statistics

For a detailed description of the MODIFY command and the Apply program refer to “OS/390 Operator Commands for the Apply Program” on page 390.

IMS Apply program configuration

IMS GEN

The IMS Apply program is an application BMP. It must be defined in the IMS system (target system) just as any other application BMP. That is, the APPLCTN macro is used. For example:

```
APPLCTN PSB=MYQAPPL,PGMTYPE=BATCH,SCHDTYP=PARALLEL
```

The PSB name can be any valid PSB name. You need to define one IMS APPLY program for each MQ Queue you have to read. One instance of IMS Apply can only read from one MQ Queue. With the SCHDTYP set to parallel you could use the same PSB for more than one instance of IMS Apply. But if one fails and the PSB is stopped it might affect an unrelated IMS Apply instance. Use one APPLCTN macro for each concurrent IMS Apply that is needed.

All of the databases that are to be accessed by this IMS system must be defined by using the DATABASE macro. ACCESS=UP is recommended. The minimum access level must be at least UPDATE.

ERROR DB -- An error database EKYERRDB must also be defined to the IMS GEN. Use the macro `DATABASE DBD=EKYERRDB,ACCESS=UP`. Refer to the EKYUIAPE sample job located in library *hlq*.SEKYSAMP for DBD source definition information.

PSB

Because IMS Apply is an application program, a PSB is required. The PSB is a normal IMS BMP-style PSB. The PSB should have a PCB for each database for which this instance of Apply will process updates. If the MQSeries Queue has messages for three different databases, then the PSB would have to have at least three PCBs. Because IMS Apply uses the AEIB interface, the order of the PCBs is not important. Each PCB must have a label or PCBNAME= specified for it. If the target database and the source database DBD names are the same, then the DBD= and PCBNAME= parameters on the PCB would be the same. If a different DBD is used for the target, then the DBD=*targetdbd* and PCBNAME=*sourcedbd* would be different. Examples are shown in Figure 433 and Figure 434.

```
PCB TYPE=DB,DBDNAME=MYQDBD,PROCOPT=A,KEYLEN=6, PCBNAME=MYQDBD,SENSEG NAME=SEGA,PARENT=0
```

Figure 433. PCB example with the same target and source DBD names

```
PCB TYPE=DB,DBDNAME=MYQDBD,PROCOPT=A,KEYLEN=6,
PCBNAME=TARQDBD,SENSEG NAME=SEGA,PARENT=0
```

Figure 434. PCB example with the different target and source DBD names

If the source and target DBD names are not the same, they must be identical in the structure of the database. That means the segment names and order must be the same. If the database is HDAM, the randomizer parameters could be different. Physical data set characteristics are not important.

In addition to the application PCBs, IMS Apply requires one additional DB PCB. This is for an error database. IMS Apply uses this database to place any MQ messages read which are in error for some reason. The placing of these messages into this database is an option which can be specified on the IMS Apply control cards. If the FAILURES control card is specified for this instance of IMS Apply, then those errors that match the CATEGORY specified will be written to the error database. If the FAILURE control card is not present, then any failure of IMS Apply to update the target database will cause the IMS Apply program to ABEND and terminate. The MQ messages that are not committed will be backed out and left on the queue.

Special IMS Apply program database: The error database must be defined as shown in Figure 435 on page 582. The DBD name is not changeable. All IMS Apply PSBs must use the same DBD name.

Recommendation: If you are in a data sharing environment, this database should be shared between all the IMS systems in the group. Whether this database is registered to DBRC is optional from an IMS Apply perspective.

```
EKYERRDB PCB TYPE=DB,DBDNAME=EKYERRDB,PROCOPT=A,KEYLEN=34, SENSEG NAME=EKYIESEG,PARENT=0
```

Figure 435. Specifying the PCB for the required error database EKYERRDB

The only other entry in the PSB is the PSBGEN statement. The PSB must be coded as `LANG=ASSEM,CMPAT=YES`, as shown in Figure 436. The PSBNAME is defined based on the IMS Gen input.

```
PSBGEN PSBNAME=MYQAPPL,LANG=ASSEM,CMPAT=YES
```

Figure 436. PSB example

Because the PSB for Apply must have a PCB for each database for which it might receive updates, the design of the PSBs implies that you understand which MQ queues contain updates for which IMS databases. There should be some logical structure to the way that you place database updates into the MQ queues.

ACB

As with any PSBs and DBDs, they need to have been generated as ACBs in the ACBLIB and placed in the active ACBLIB.

JCL

The JCL for IMS Apply is the same as that of any application BMP. Normal rules apply. If the security gen is protecting AGNs, then you will need to treat this BMP as you do all other BMPs. The PSB you specify is the one defined to the IMS systems as usual. The program resides in the EKYRESLB library. Sample JCL is shown in Figure 437 on page 583.

```
//*-------------------------------------------------------------------------*
//* RUN THE DPROP IMS-APPLY AS IMS BATCH *
//*-------------------------------------------------------------------------*
//BMP PROC PROG='',
//    PSB='',
//    SOUT='*'
//PUTMSG EXEC PGM=DFSRRC00,REGION=4096K,
//PARM=(BMP,&PROG,&PSB,,,C00000,,1,,,,,,&IMSID,&AGN)
//STEPLIB  DD DSN=22222222.EKYRESLB,DISP=SHR
//         DD DSN=11111111.RESLIB,DISP=SHR
//DFSRESLB DD DSN=11111111.RESLIB,DISP=SHR
//IMS      DD DSN=33333333.DBDLIB,DISP=SHR
//         DD DSN=33333333.PSBLIB,DISP=SHR
//PROCLIB  DD DSN=11111111.PROCLIB,DISP=SHR
//EKYRESLB DD DSN=22222222.EKYRESLB,DISP=SHR
//DFSESL   DD DSN=11111111.RESLIB,DISP=SHR
//         DD DSN=44444444.MQLIBRARY,DISP=SHR
//PEND
//* END OF BMP PROC
//A EXEC BMP,
//    IMSID='IMSID',
//    AGN='YOURAGN',
//    PSB='YOURPSB',
//    PROG='EKYMQIAP'
//EKYIN    DD DUMMY
//APPLYIN  DD * APPLY NAME=aaaaaaaa; QMANAGER NAME=mmmm; QUEUE NAME=qqqqqqqq;
//EKYPRINT DD SYSOUT=*
//EKYTRACE DD SYSOUT=*
//EKYWTO   DD SYSOUT=*
//EKYSNAP  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

Figure 437. IMS Apply JCL sample

Apply must run on the same MVS system as the MQ system from which it is reading the queues.

The AGN and IMSID parameters are the same as for any BMP. The PSB name is the PSB to be used by this instance of IMS Apply. The program name is always EKYMQIAP.

The EKYTRACE DD statement provides trace information to debug a problem. It should normally be DUMMY or a null file unless you are trying to resolve.

The DFSESL DD statement needs to contain the MQ library for this MQ system.

The //APPLYIN DD contains the following input parameters for this instance of Apply.

APPLY NAME=

APPLY NAME= is designed to identify an instance of Apply. This is a user-defined name and can be any 8 characters. Because the MQ queue must be read sequentially, and because the updates are processed in the sequence in which they are in the queue, use the queue name as the APPLY name. Then enforce that only Apply is running at the same time.

After the Apply program is started, it will continue to process until it is specifically stopped. It will process new messages as they arrive on the queue. To stop the APPLY program, issue a `/P` *`jobname`* `MVS` command from SDSF or any other location were you can issue MVS commands.

QMANAGER NAME=
The QMANAGER NAME= is the name of the MQ system on this MVS system that has the specified queue.

Queue NAME=
This is the name of the MQSeries queue that will be read.

Note: You can specify additional control statements to the //APPLYIN file.
Five sample jobs are available that you can customize to run the IMS Apply program and verify correct functioning. These supplied jobs are designed to be used in conjunction with the MQ Capture IVP jobs that are supplied with IMS DPROP, which are customized by using the product-supplied installation and customization panels. These supplied sample IMS MQ Apply jobs are designed to process a workload captured by a corresponding MQ Capture system driven by the ISPF IVP panel application.

Notes:

1. It is necessary to generate an IMS DPROP MQ Capture system associated with capturing of IMS changed data. Refer to the existing product publications for information on establishing the MQ Capture environment.
2. There is no requirement to generate an IMS DPROP system associated with the IMS Apply process.

The five new sample jobs are located in IMS DPROP product library *hlq*.SEKYSAMP, members EKYUIAPE, EKYUIAPT, EKYUIAPP, EKYUIAPJ, and EKYUIDSP. The jobs are designed to be edited, customized, and submitted in the following sequence:

EKYUIAPE
IMS ERROR DB housekeeping, DBD, allocate and initial load

EKYUIAPT
IMS IVP target DB housekeeping, allocate and initial load

EKYUIAPP
PSB for IMS Apply

EKYUIAPJ
IMS Apply IVP job

EKYUIDSP
Simple display of IMS source and target databases

Notes:

1. In member EKYUIAPJ, the IMS Apply program is set up to run in DLI mode simply for convenience.
2. To ensure data integrity, the IMS Apply program must be run as a BMP.

Chapter 23. IMS-to-IMS replication Selector component

If you already use the log-based asynchronous DPROP features for IMS-to-DB2 propagation, you are familiar with Selector processing. To implement IMS-to-IMS propagation you must also become familiar with MQ-async propagation and with the IMS Apply program described in “The IMS Apply program” on page 569. If you are not already familiar with the MQSeries-based features of DPROP, you will also need to understand the Capture program. See Chapter 14, “Capture Component,” on page 363 for more information about the Capture program.

The standard Selector component collects propagation log records from IMS log files and writes them to propagation request data sets for later processing by the Receiver component. In support of IMS-to-IMS propagation, the Selector is extended to capture the IMS changes that are found within these IMS log records and to write them to MQSeries messages for later processing by the IMS DPROP-supplied IMS Apply program.

Refer to Chapter 10, “The Selector,” on page 313 for detailed information about the Selector component and its setup. The differential information that is required for the Selector for IMS-to-IMS replication to create MQSeries messages similar to the IMS Capture program is described in “SELECT control statement.” The SELECT control statement of the //EKYSIDS file was extended to provide support for IMS-to-IMS replication.

Refer to *IMS Data Propagator for z/OS Administrators Guide for MQSeries Asynchronous Propagation* for more information about MQSeries messages.

The following topics provide additional information:

- “SELECT control statement”

SELECT control statement

To indicate to the Selector that you want it to write data capture changes to MQSeries for subsequent processing by the IMS Apply program, a QMGR, and an associated QUEUE subcommand must be provided by using the SELECT control statement.

Figure 438 on page 586 shows the syntax of the extended SELECT control statement.

```
                    ┌─,──────┐
                    ▼        │
►►─SELECT─┬─GROUP=(──group_id──)─┬──────────────────────────────────────────────►◄
          └─ALL──────────────────┘  │                                            │
                                    └─,─STOP─=─┬─INTERIM──────────┬─┘
                                               ├─timestamp────────┤
                                               └─TSM,ID=tsm_id────┘

►►─┬──────────────────────────────────┬─;──────────────────────────────────────►◄
   │                                  │
   └─,─INVUOW─=─┬─STOP───┬─┘
                └─IGNORE─┘

►►─┬──────────────────────────────────────────────────────────────────────────►◄
   └─,─QMGR─=─queue_manager name─,─QUEUE─=─queue_name─┘
```

Figure 438. The SELECT control statement

QMGR=name_of_mqseries_queue_manager

The QMGR subcommand is mandatory when the Selector is used to capture IMS data changes for IMS-to-IMS replication. In this instance, you must provide only one QMGR=*keyword* for each Selector issued.

The QMGR subcommand identifies which MQSeries Queue Manager is used to send the changed data to the Apply component.

QUEUE=name_of_mqseries_input_queue

The QUEUE subcommand identifies the name of an MQSeries queue (or the name of an MQSeries queue alias) that has been reserved for use by the Selector. The Selector writes the propagation data into the MQSeries queue that is identified on the QUEUE=*keyword*.

When the Selector is run, the MQSeries queue or alias identified here must exist and be ready for use.

Multiple SELECT control statements can be specified for one Selector processing, and each SELECT control statement can specify a different QUEUE name so that different GROUPs can have captured changes directed to different MQSeries queues. Conversely, you can choose to direct captured changes from different groups to the same MQSeries queue by specifying the same QUEUE name for different GROUPs. If multiple GROUPs specify the same queue name, all of the data of all the specified GROUPs flow in FIFO (first in, first out) sequence to the IMS Apply program.

COMMIT_IVAL=n

The COMMIT_IVAL subcommand specifies the number of committed source units of work to write to MQSeries before issuing an MQSeries commit.

1 Specifies that an MQSeries commit is to be issued for every committed source unit of work written to MQSeries. This value is the default for COMMIT_IVAL.

n Specifies that an MQSeries commit is to be issued after *n* committed source units of work are written to MQSeries, where *n* is an integer in the range of 1 to 99999999.

SELECT control statement examples

Example 1 – SELECT with IMS data changes for selected groups sent to one MQSeries queue

The example in Figure 439 sends changes selected for the GROUP01 and GROUP2 propagation groups to one MQSeries queue.

```
SELECT GROUP=(GROUP01,GROUP02),QMGR=MQD1,QUEUE=MQ.APPLY.QUEUE ;
```

Figure 439. SELECT control statement for IMS-to-IMS replication, example 1

Example 2 – SELECT with IMS data changes for selected groups sent to two MQSeries queues

The example in Figure 440 sends changes selected for the GROUP01 propagation group to one MQSeries queue and GROUP2 propagation group to a another (different) MQSeries queue.

```
SELECT GROUP=GROUP01,QMGR=MQD1,QUEUE=MQ.APPLY.QUEUE01 ;
SELECT GROUP=GROUP02,QMGR=MQD1,QUEUE=MQ.APPLY.QUEUE02 ;
```

Figure 440. SELECT control statement for IMS-to-IMS replication, example 2

Example 3 – SELECT with all IMS data changes sent to one MQSeries queue

The example in Figure 441 sends changes for all propagation groups in the SCF one MQSeries queue.

```
SELECT ALL,QMGR=MQD1,QUEUE=MQ.APPLY.QUEUE ;
```

Figure 441. SELECT control statement for IMS-to-IMS replication, example 3

Part 10. Appendixes

Appendix A. Coding the Description

Several IMS DPROP commands have an optional keyword DESC= *comment*. The rules for coding that keyword are:

- The description can include purely character data, or a mixed string of character and DBCS data.
- The DBCS portion of a mixed string must be bracketed by a shift-out (X'0E') and shift-in (X'0F') character, as in this example:

  ```
  'characterdata<D_B_C_S>'
  ```

  (X'0E' is represented by < andX'0F' by >.) A maximum of 50 bytes is allowed for both character and mixed descriptions.
- Enclose descriptions, character or mixed, in single quotes. If single quotes are part of the stored text, double each quote so that it will not be misinterpreted as a string terminator.

Appendix B. Output Data Records and Fields

This topic defines a data record, describes its role in an output job, and details how individual fields are represented.

The following topics provide additional information:
- "What is a Data Record"
- "How Individual Fields are Represented in EBCDIC Format" on page 594
- "How Fields are Represented in Source Format" on page 599
- "How IXF Records are Represented " on page 600

What is a Data Record

A data record is a line of data in a file. Data records either replace the *EO statement in the JCS associated with your extract request or are inserted in the data set named on the EXTDATA keyword of your request. As such, they represent the extracted data.

The extracted data, or data records, can be in either SOURCE or EBCDIC format. Use the FORMAT keyword of the SUBMIT command to specify the format you require.

In the data records, each extracted field is preceded by a one-byte null indicator. If a particular value in a field is null, the byte contains a hyphen. If the value is not null, the byte is blank.

Data records also represent messages if the JCS you use for loading your extracted data can also be used for loading IMS DPROP messages into a message table.

Data Records in an Output Job

When your extracted data is embedded in an output job, each data record represents all or part of a row of the target table or message table. All the data records embedded in an output job (to send data across networks) are 80 characters (bytes) long. If all the data in a given row cannot be represented by a single data record, the row is spread across a succession of additional data records:

- Within a data record for a given table row, characters 2 through 80 can represent data.
- If more than one data record is needed, columns 2 through 80 on the additional records can also represent data; column 1 never represents data.
- Column 1 of a data record is used to indicate continuation. It carries a hash sign (#) if it and the next record represent parts of the same table row; otherwise, it contains a blank.

In the example below, the first column of the first record contains a pound sign; one table row, therefore, consists of two data records.

```
# 10 HEAD-OFFICE  160 CORPORATE  NEW YORK  160 MOLINARE
  MGR  000007 0229 59.20 000000.00
```

Figure 442. IMS DPROP System Overview Pop-up Panel

When more than one record for each row is needed, fields can wrap around from one record to the next. The wrap-around is continuous. For example, a 4-byte field that begins in column 79 of one record has its first 2 bytes in columns 79 and 80 of that record, and its last 2 bytes in columns 2 and 3 of the next record.

Data Records in Data Sets and Files

When your extracted data is written to a physical sequential data set or CMS file, each data record represents a row of the target table or message table. Data records in data sets or files can be any number of characters (bytes) long, depending on the lengths of the fields being extracted. Within a data record for a given table row, all the characters (that is, characters 1 through the last character) can represent data.

How Individual Fields are Represented in EBCDIC Format

This topic gives a detailed account of how Data Refresher fields are represented in IMS DPROP output when the fields are converted from source to EBCDIC format.

source field For a given row of output, the source fields are the fields in the source or sources of data that are listed in the SELECT keyword of the EXTRACT statement.

target field A target field is the representation of some source field in a data record.

Within a data record, the target fields for a row of output appear in the order in which their source fields are named in the SELECT statement of the corresponding EXTRACT statement.

There is no space between target fields. The first target field for a row begins in column 2 of its data record, the second immediately follows the first, and so on.

A target field begins with a one-column *nulls indicator*:

- If the nulls indicator contains a hyphen (-), the target field is null (value unknown) and is set to blank.
- If the column is blank, the target field is not null.

Because most fields are not null, the nulls indicator also acts as a separator between columns when the output data set or file is printed or is displayed at your terminal.

The data portion of a target field follows the nulls indicator. When the field is not null and you have indicated that the source data should convert to EBCDIC format, the data portion represents the data as an EBCDIC character string, no matter what the data characteristics of the source field. If, for example, the source field is an H-type field with a hexadecimal content of X'8000', the data portion of the target field would contain -32768, which is the decimal representation of the integer contained in the source field. When the nulls indicator contains a hyphen, the data portion of the field is set to blanks.

Note: If a target field is to be loaded into a column and if the source field is zoned or packed decimal, assign a value other than (SUBST(NULL)) or (SUBST(NULL),n) to the FLDERR keyword on the OPTIONS keyword of the EXTRACT statement. With either of these two options, IMS DPROP substitutes nulls and generates NULL IF statements for the fields in your extract request that contain errors; this would result in the extract request being rejected.

The following topics discuss how the data portion of a target field represents its data in EBCDIC format. This depends on the data characteristics and length of the source field.

B-Type Fields

When the source field is a B-type field, it represents an integer as a one-byte (8-bit) unsigned binary integer in the range of 0 through 255. Since the higher order bit is interpreted as data, not as the sign, B-type fields are always positive. When extracted for a relational or target in IXF, B-type fields map to the corresponding two-byte signed binary (H-type) columns or fields in the target.

C-Type Fields

When the source field is a C-type field, the data portion of the target field is merely a copy of the source field. Normally, a byte in the source field (and its copy in the target field) has an EBCDIC representation. However, this need not be true; a byte in a C-type field can have any one of the possible 256 values, whether or not that value represents an EBCDIC character.

A-Type, T-Type, and S-Type fields

A-type (date) data, T-type (time) data, and S-type (timestamp) data is collectively known as *date/time data*.

Note: When extracting date/time data with the DEM, the DEM produces date/time output in standard ISO format. This is not the case when using the REM to extract data. User-supplied date/time conversion exits used in the DEM are also expected to produce date/time data in standard ISO.

The standard ISO format for each type of date/time data output is shown in the following table:

Data Type	Format	Description
DATE	yyyy-mm-dd	A date is always a ten-byte character string
TIME	hh.mm.ss	A time is always an eight-byte character string
TIMESTAMP	yyyy-mm-dd-hh.mm.ss.nnnnnn	A timestamp is always a 26-byte character string

where:
yyyy = year
mm = month
dd = day
hh = hour
mm = minute
ss = second

nnnnnn
= nanosecond

```
1986-12-30                  (Date data)    03.59.30      (Time data)
03.59.30                    (Time data)
1963-07-29-11.59.58.000001  (Timestamp data)
```

Figure 443. Timestamp Examples

D-Type and E-Type Fields

This topic illustrates how source fields are represented when they are D-Type and E-Type.

D-Type Fields

When the source field is a D-type field, it represents a number in long (8-byte) floating point format. Its target field represents this number in the E-form (that is, exponent form) used by PL/I. The data portion of this field is 23 bytes long. It contains a mantissa and an exponent, which are separated by an uppercase 'E'.

- The mantissa, to the left of the E, is a 17-digit, signed number with a decimal point between its first and second digits. Unless the number represented is zero, the leftmost digit in the mantissa is never zero. If the number is negative, this digit is preceded by a hyphen. If it is positive, the digit is preceded by a leading zero.
- The exponent is a two-digit, signed integer that indicates how many places to the right or left the decimal point in the mantissa must be moved to obtain the number represented. A positive value indicates movement to the right. A negative value indicates movement to the left. A value of zero implies that the number represented is the mantissa.

Examples:

```
-1.2345678901234567E+10 represents -12345678901.234567    1.2345678901234567E-03 represents  .0012345678901234567
 0.0                    represents  0
```

Figure 444. D-Type Field

Note: Zero, as shown above, has a special representation that does not follow the rules stated above. All other numbers have representations that do follow the rules.

E-Type Fields

When the source field is an E-type field, it represents a number in short (4-byte) floating point format. As with a D-type field, its target field represents this number in E-form notation, but the representation is shorter. The data portion for an E-type field is only 14 bytes long to accommodate an eight-digit mantissa.

Examples:

```
-1.2345678E+04  represents -12345.678    1.2345678E-3   represents  .0012345678
 0.0            represents  0
```

Figure 445. E-Type Field

F-Type and H-Type Fields

This topic illustrates how source fields are represented when they are F-type and H-type.

F-Type fields

When the source field is an F-type field, it represents an integer as a fullword (32-bit) signed binary integer. The target field represents this number in decimal. Its data portion is 11 bytes long to accommodate the largest negative integer (including sign) that the source field can represent. A hyphen always appears at the left edge of the data field if the number represented is negative. If the number is positive, this leftmost byte contains a leading zero.

Examples:

```
0000012345   represents the integer 12345.   -2147483648   represents the largest negative
             integer that the source field
             can hold.
```

Figure 446. F-Type Field

H-Type Fields

When the source field is an H-type field, it represents an integer as a halfword (16-bit) signed binary integer. The target field represents this integer in decimal exactly as it would for an F-type field, except that the data portion of the field is only 6 bytes long.

Examples:

```
 00123   represents the integer 123.
-32768   represents the largest negative integer
         that the source field can hold.
```

Figure 447. H-Type Field

G-Type Fields

When the source field is a G-type field, it represents double-byte character set (DBCS) data. This means that two bytes are used to represent one DBCS character. The length is specified in bytes, and must be an even number in the range of 2 through 254. The DBCS data input cannot be delimited by shift-out (X'0E') and shift-in (X'0F') characters in storage.

If the INTO keyword is omitted from the EXTRACT command, IMS DPROP assumes that the data is targeted for a physical sequential data set or CMS file. In

this case, when IMS DPROP extracts data from a G-type field, shift-out and shift-in characters are included for all DBCS data. The source data from a G-type field is assumed to be totally DBCS data (not mixed).

For relational databases, the shift-out and shift-in characters are not included with DBCS data, because SQL/DS™ and DB2 assume that G-type fields are totally DBCS data and not mixed. If DBS=IXF, shift-out (X'0E') and shift-in (X'0F') characters are placed around all extracted GRAPHIC data, even if an INTO keyword is specified.

P-Type and Z-Type Fields

This topic illustrates how source fields are represented when they are P-Type and Z-Type.

P-Type Fields

When the source field is a P-type field, it represents a number in packed decimal format. The target field represents this number in fixed decimal format; that is, as a sign followed by a string of digits and a decimal point.

Example:

```
-123.45
```

The decimal point can lie within the string of digits or immediately to the right or left of it, the position having been determined by the scale attribute of the source field. The scale attribute is a non-negative integer that indicates how many decimal digits lie to the right of the decimal point. For example, the source field represented by -123.45 has a scale attribute of 2.

The scale attribute of the source field is contained in the field's underlying IMS DPROP file or PSB description. It was specified by the FIELD keyword that defined the source field.

The data portion of the target field is just large enough to hold the largest negative number that the source field can represent. For an *n*-byte source field, this requires 2x *n*+1 bytes. Non-significant leading zeros are represented as zeros, instead of blanks. If the number represented is negative, the leftmost byte in the data portion of the target field contains a hyphen. If the number is positive, this byte contains a leading zero.

Examples:

```
The target field for a 3-byte source field
might contain:
 -12345. (scale factor of zero)
  005.43 (scale factor of two)
  .02178 (scale factor of five)
```

Figure 448. P-Type Field

Z-Type Fields

When the source field is a Z-type field, it represents a number in zoned decimal format. The target field represents this number in fixed decimal format, just as it would if the source field were a P-type field. The representation differs from that of a P-type field only in the relative lengths of the source and target fields. Because a

Z-type field represents one decimal digit for each byte instead of two, the data portion of the target field for an *n*-byte, Z-type field is *n*+2 bytes long.

VC-Type and VG-Type Fields

This topic illustrates how source fields are represented when they are VC-Type and VG-Type.

VC-Type Fields

When the source field is a VC-type field, the data portion of the target field is merely a copy of the variable-length source field. Normally, a byte in the source field (and its copy in the target field) has an EBCDIC representation. However, this need not be true; a byte in a VC-type field can have any one of the possible 256 values, whether or not that value represents an EBCDIC character.

VG-Type Fields

When the source field is a VG-type field, it represents variable-length double-byte character set (DBCS) data. This means that two bytes are used to represent one DBCS character. For a VG-type field, give the length in an even number of bytes from 2 through 254. DBCS data cannot be delimited by shift-out (X'0E') and shift-in (X'0F') characters in storage.

When the INTO keyword is omitted from the EXTRACT command, IMS DPROP assumes that the data is targeted for a physical sequential data set or CMS file. In this case, when IMS DPROP extracts data from a VG-type field, shift-out and shift-in characters are included for all DBCS data. The source data from a VG-type field is assumed to be all DBCS data (not mixed).

For relational databases, the shift-out and shift-in characters are not included with DBCS data, because SQL/DS and DB2 assume that VG-type fields are totally DBCS data and not mixed. If DBS=IXF, shift-out (X'0E') and shift-in (X'0F') characters are placed around all extracted GRAPHIC data, even if an INTO keyword is specified.

When extracted for a relational or target in IXF, VC-type and VG-type fields map to the corresponding variable-length columns or fields in the target.

A VC-type or VG-type field can have an accompanying length field of type F, H, B, P, or Z (zero scale for P or Z) that specifies the size in bytes of the VC-type or VG-type field. The length field is coded on the LFIELD keyword in the IMS DPROP file or DXTPSB description.

The BYTES keyword, coded in the IMS DPROP file or DXTPSB description, specifies the maximum size of a VC-type or VG-type field. If not indicated by a length field, the size of a VC-type or VG-type field is computed as the record or segment length plus one minus the starting position of the VC-type or VG-type field.

How Fields are Represented in Source Format

When you request that IMS DPROP extract data and keep it in source format (rather than convert it to EBCDIC), the data is extracted and presented in exactly the same way as it was entered in source. There are, however, three exceptions to this rule:

- Zoned decimal fields are *always* converted to packed decimal.
- When the target database is SQL/DS, single precision floating point fields are converted to double precision floating point.

- Date, time, and timestamp type fields are always output in full character ISO (if you are using the DEM to extract data), regardless of the format specified on the EXTRACT statement of the SUBMIT (UIM) command. (See “A-Type, T-Type, and S-Type fields ” on page 595 for the definition of full character ISO.)

How IXF Records are Represented

There are seven different record types defined for IXF. IMS DPROP generates the four types supported in IXF Version 0, the first IXF implementation. Three of these record types are informational or descriptive. They define the file in IXF and the data it contains. These records are always in character format, and numeric values are right-justified.

The data itself can be in character or machine (mixed binary and character) format. DBCS data is not converted to character format.

Appendix C. Data Propagation Examples

This topic provides examples of the various steps involved in data propagation. These steps include:

- Defining DBDs
- Defining the propagated DB2 tables
- If you are using DataRefresher to define your propagation requests and to extract the IMS data:
  - Defining DXTPSBs and DXTVIEWs
  - Coding DataRefresher SUBMIT commands and EXTRACT statements
  - Providing DataRefresher DEM control statements
- If you are using the MVGIN Front End Application to define your propagation requests:
  - Interacting with the ISPF application
  - Providing MVGU control statements
- If you are using SQL insert statements to define your propagation requests:
  - Defining your propagation requests in the MVG input tables
  - Providing MVGU control statements

This topic provides the following examples:

1. "Example 1: Propagation with Mapping Case 1 " on page 603 describes the definitions required to propagate one segment type to one DB2 table with a propagation request belonging to mapping case 1.

   This includes how to:

   - Define a propagation request belonging to mapping case 1.
   - Define the mandatory primary DB2 key of the target table.
   - Propagate the fully concatenated key of the entity segment to the columns of the DB2 primary key. You do this by propagating a set of key subfields that completely overlay the fully concatenated key.
   - Propagate a packed IMS field to a decimal DB2 column. This example shows how to express the length of a packed IMS field in bytes, while the length of the decimal DB2 column is expressed in digits.
   - Map a graphic IMS field to a graphic DB2 column. This shows how to express the length of a graphic IMS field in bytes, while the length of the DB2 column is expressed in DBCS characters.
   - Map a variable-length graphic IMS field to a variable-length graphic DB2 column. This example describes the definitions associated with the length field of a variable-length field.
2. "Example 2: Propagation with Mapping Case 2 " on page 621 describes the definitions required to propagate an entity segment type and two extension segment types to one DB2 table with a propagation request belonging to mapping case 2.

   This includes how to:

   - Define a propagation request belonging to mapping case 2.
   - Define the mandatory primary DB2 key of the target table.
   - Propagate the fully concatenated key of the entity segment to the columns of the DB2 primary key. You do this by propagating a set of key subfields that completely overlay the fully concatenated key.
   - Propagate a packed IMS field to a decimal DB2 column. This example shows how to express the length of a packed IMS field in bytes, while the length of the decimal DB2 column is expressed in digits.

3. “Example 3: Propagation with Mapping Case 3 ” on page 643 describes the definitions required to propagate repeating group of fields with ***mapping case 3***. In this example, the propagated IMS segment contains two repeating groups of fields propagated by two mapping case 3 propagation requests to two different tables.

   This includes how to:

   - Propagate the two repeating group of fields (internal segments) to two different tables with propagation requests belonging to mapping case 3.
   - Propagate fields that are not part of the repeating group of fields (the fields of the containing segment) to another table with a propagation request belonging to mapping case 1.
   - Define the mandatory DB2 primary key of the three target tables.
   - Propagate the combination of the ID fields of the internal segments and the fully concatenated key of the containing segment to the columns of the DB2 primary key of the target table of the mapping case 3 propagation requests.
   - Propagate the fully concatenated key of the containing segment to columns of the DB2 primary key of the target table of the mapping case 1 propagation request.
4. “Example 4: Propagation with a WHERE Clause and PATH data: 4 ” on page 679 describes an example of propagation request definitions with a ***WHERE clause*** and ***PATH data***.
5. “Example 5: Propagating a Complete IMS Database Structure ” on page 704 is a more complex example. It describes the propagation of all the segments of an IMS database to DB2 tables with propagation requests belonging to mapping case 1 and mapping case 2.

   This includes how to:

   - Define propagation requests belonging to mapping case 1 and propagation requests belonging to mapping case 2.
   - Implement DB2 referential integrity relationships (RIRs) between the propagated tables, which match the parent/child relationships in the IMS database.
   - Define the DB2 primary keys of the DB2 tables.
   - Map IMS fields belonging to the fully concatenated key of the entity segments to columns of the DB2 primary key.
   - Define DB2 foreign keys.
   - Map IMS fields belonging to the concatenated key of the propagated entity segment to columns of the DB2 foreign key.

Additional examples of propagation request definitions involving Segment exit routines, Field exit routines, and Propagation exit routines are in IMS DPROP Customization Guide.

Table 46 shows the naming conventions for databases, segments, tables, fields, and columns apply to most examples:

Table 46. Naming Conventions for Databases, Segments, Tables, Fields, and Columns

Names beginning with:	Represent:	Examples:
D	databases	D01 D02
S	segments	S11 S12

Table 46. Naming Conventions for Databases, Segments, Tables, Fields, and Columns (continued)

Names beginning with:	Represent:	Examples:
T	tables	T11 T12
F	fields	F11K011 F11D022
C	columns	C11K011 C11D022

Note:
1 K stands for key
2 D stands for data

The following topics provide additional information:
- "Displaying the Function Keys and Selecting Options "
- "Example 1: Propagation with Mapping Case 1 "
- "Example 2: Propagation with Mapping Case 2 " on page 621
- "Example 3: Propagation with Mapping Case 3 " on page 643
- "Example 4: Propagation with a WHERE Clause and PATH data: 4 " on page 679
- "Example 5: Propagating a Complete IMS Database Structure " on page 704

Displaying the Function Keys and Selecting Options

If you want to display the function keys for the panels, type PFSHOW on the command line of any panel and press Enter. The function keys will be shown at the bottom of the screen until the end of the session or until you type PFSHOW OFF on the command line and press Enter.

The examples in this topic frequently tell you to select a specified option. You can select an option in either of the following ways:
- Type the number of the required option on the command line and press Enter.
- Position the cursor to the left of the required option and press Enter.

Example 1: Propagation with Mapping Case 1

Figure 449 on page 604 shows the propagation of segment S02 of the following database model. For this propagation, use a propagation request belonging to mapping case 1.

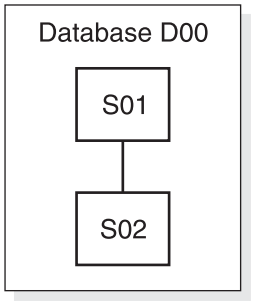

Figure 449. Propagating a Mapping Case 1 Propagation Request (Example 1)

This example describes the definitions to propagate:
- the fields of the fully concatenated key of S02
- data fields stored in S02

Overview of the Propagation Performed: 1

Figure 450 illustrates the propagation of segment S02, by PR2, to table T02.

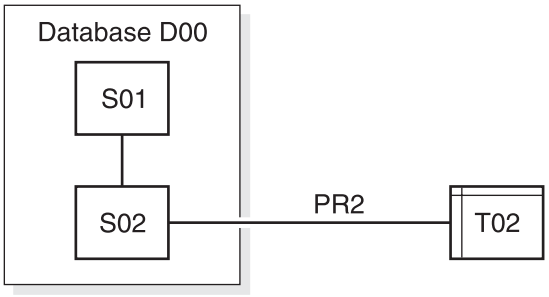

Figure 450. Overview of Propagation of Database D00 (Example 1)

Figure 451 on page 605 illustrates the mapping between the fully concatenated key of segment S02 and the DB2 primary key of table T02.

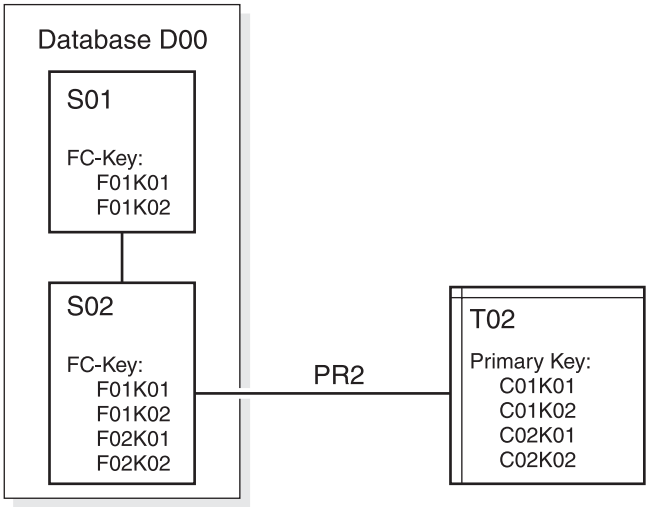

Figure 451. Key Mapping between D00 and T02 (Example 1)

Notes::

1. The fully concatenated key of the entity segment illustrates propagation, as IMS DPROP recommends, to the DB2 primary key.

   The fully concatenated key of S02 consists of the key field of S01 (the F01KEY field defined in the IMS DBD) and the key field of S02 (the F02KEY field defined in the IMS DBD).
2. The fully concatenated key of S02 is *not* mapped by propagating the two key fields F01KEY and F02KEY to two columns of the DB2 primary key. Instead, the fully concatenated key is propagated to the DB2 primary key by mapping 4 subfields of F01KEY and F02KEY to 4 columns of the DB2 primary key:
   - The subfield of F01KEY, F01K01, is mapped to C01K01
   - The subfield of F01KEY, F01K02, is mapped to C01K02
   - The subfield of F02KEY, F02K01, is mapped to C02K01
   - The subfield of F02KEY, F02K02, is mapped to C02K02

Field Mapping Overview: 1

Table 47 on page 606 shows the characteristics of the source IMS fields and the target DB2 columns. The table contains the following information:

- For the IMS source field:
  - The name of the IMS field (as defined to IMS DPROP either on the NAME keyword of the DataRefresher FIELD statement, or in the FLDNAME column of the DPRIFLD MVG input table)
  - The data type (as defined to IMS DPROP either on the TYPE keyword of the DataRefresher FIELD statement, or in the DATATYPE column of the DPRIFLD MVG input table)
  - The length in bytes (as defined to IMS DPROP either on the BYTES keyword of the DataRefresher FIELD statement or in the BYTES column of the DPRIFLD MVG input table)
- For the DB2 target column:
  - The name of the target DB2 column (as defined to DB2 in the CREATE TABLE statement)
  - The data type of the target DB2 column (as defined to DB2 in the CREATE TABLE statement)

Table 47. Source Segment: S02 (and its concatenated key) - Target Table: T02

Source IMS fields				Target DB2 columns	
Field name	**See Note Below**	**Data Type (DXT)**	**Length (in Bytes)**	**Column Name**	**Data Type**
F01K01	1	C	2	C01K01	CHAR(2)
F01K02	1	C	3	C01K02	CHAR(3)
F02K01	2	C	2	C02K01	CHAR(2)
F02K02	2	C	4	C02K02	CHAR(4)
F02D03		B	1	C02D03	SMALLINT
F02D04	3	P	2	C02D04	DECIMAL(3,0)
F02D05		Z	3	C02D05	DECIMAL(3,0)
F02D06		C	10	C02D06	CHAR(10)
F02D07	4	H	2	-	
F02D08	5, 6	VG	100	C02D08	VARGRAPHIC(50)

Notes::

1. F01K01 and F01K02 are located in the concatenated key of the entity segment. They are subfields of the key field of parent segment S01.
2. F02K01 and F02K02 are subfields of the key field of entity segment S02. They are propagated to columns of the DB2 primary key of T02.
3. F02D04 is a packed IMS field. Its length is defined to IMS DPROP in bytes (2). C02D04 is the target column, and its length is defined to DB2 (in the CREATE TABLE statement) in digits (3).
4. F02D07 is the length field of the variable-length graphic field F02D08. In this example, as in most real-life cases, the length field is not propagated and therefore has no target DB2 column.
5. F02D08 is a variable-length graphic field. Its length field is F02D07.
6. F02D08 is a graphic IMS field. Its length is defined to IMS DPROP in bytes (100). C02D08 is the target DB2 column, and its length is defined to DB2 (in the CREATE TABLE statement) in DBCS characters (50).

DBD Definitions: 1

Figure 452 shows the DBD control statements for the model database.

```
         DBD NAME=D00,VERSION=V123456789,                            C
               ACCESS=(HDAM,OSAM),RMNAME=(DFSHDC40,5,4)
         DATASET DD1=HDAM,SIZE=4096,DEVICE=3380
*
         SEGM  NAME=S01,PARENT=0,BYTES=60
         FIELD  NAME=(F01KEY,SEQ,U),BYTES=5,START=1
*
         SEGM  NAME=S02,PARENT=S01,BYTES=(126,26),EXIT=(EKYRUP00)
         FIELD  NAME=(F02KEY,SEQ,U),BYTES=6,START=3
*
         DBDGEN
         FINISH
         END
```

Figure 452. DBD control statements (Example 1)

The EXIT keyword in the SEGM control statement tells IMS to give control to EKYRUP00, the IMS DPROP-provided DL/I Capture Exit routine, when a DL/I call updates an occurrence of S02.

You can see that in the DataRefresher EXTRACT statement or the SPUFI statements used to define the propagation request in the MVG input tables, the fully concatenated key of S02 is not propagated by mapping the key fields F01KEY and F02KEY. Instead, the fully concatenated key is propagated (to the DB2 primary key) by mapping subfields of F01KEY and F02KEY.

Usually, IMS DPROP and DataRefresher do not require you to define the non-key fields in the IMS DBD. In this example, the non-key fields are not defined in the IMS DBD.

Table Definitions: 1

Figure 453 describes the SQL statements to create the target tables shown in the following topics. As required by IMS DPROP, table T02 has a DB2 primary key. And, as recommended by IMS DPROP, the DB2 primary key will be the propagated fully concatenated key of the entity segment.

```
CREATE TABLE T02
            (C01K01   CHAR(2)           NOT NULL,
             C01K02   CHAR(3)           NOT NULL,
             C02K01   CHAR(2)           NOT NULL,
             C02K02   CHAR(4)           NOT NULL,
             C02D03   SMALLINT          NOT NULL WITH DEFAULT,
             C02D04   DECIMAL (3,0)     NOT NULL WITH DEFAULT,
             C02D05   DECIMAL (3,0)     NOT NULL WITH DEFAULT,
             C02D06   CHAR(10)          NOT NULL WITH DEFAULT,
             C02D08   VARGRAPHIC(50)    NOT NULL WITH DEFAULT,
          PRIMARY KEY (C01K01, C01K02, C02K01, C02K02))
       IN DU096606.PROPTS    ;

CREATE UNIQUE INDEX XN01 ON T02 (C01K01, C01K02, C02K01, C02K02)
       USING VCAT KOE ;
```

Figure 453. Table T02 (Example 1)

Defining Propagation Requests with DataRefresher: 1

Defining the DXTPSB: 1

DataRefresher requires that you describe the IMS database to DataRefresher by providing a CREATE DXTPSB command. The DataRefresher control statements for our model database are shown in Figure 454 on page 608. The control statements which are highlighted numerically (1 or 2, for example) are further described below the figure.

```
                                                          NOTE
CREATE DXTPSB    NAME=PSB00

       DXTPCB    NAME=PCB001,DBACCESS=HDAM,DBNAME=D00

         SEGMENT NAME=S01,
                 PARENT=0,BYTES=60

             FIELD    NAME=F01KEY , START=1,  BYTES=5 ,  SEQFLD=R        1
             FIELD    NAME=F01K01 , START=1,  BYTES=2 ,  TYPE=C          2
             FIELD    NAME=F01K02 , START=3,  BYTES=3 ,  TYPE=C          2

             FIELD    NAME=F01D03 , START=6,  BYTES=10 , TYPE=C
             FIELD    NAME=F01D04 , START=16, BYTES=45 , TYPE=C

         SEGMENT NAME=S02,
                 PARENT=S01,BYTES=126,FORMAT=V                          3

             FIELD    NAME=F02KEY , START=3,  BYTES=6 ,  SEQFLD=R        4
             FIELD    NAME=F02K01 , START=3,  BYTES=2 ,  TYPE=C          5
             FIELD    NAME=F02K02 , START=5,  BYTES=4 ,  TYPE=C          5

             FIELD    NAME=F02D03 , START=9,  BYTES=1 ,  TYPE=B
             FIELD    NAME=F02D04 , START=10, BYTES=2 ,  TYPE=P
             FIELD    NAME=F02D05 , START=12, BYTES=3 ,  TYPE=Z
             FIELD    NAME=F02D06 , START=15, BYTES=10,  TYPE=C
             FIELD    NAME=F02D07 , START=25, BYTES=2 ,  TYPE=H          6
             FIELD    NAME=F02D08 , START=27, BYTES=100 ,TYPE=VG,
                      LFIELD=F02D07 ;                                   7
```

Figure 454. DataRefresher control statements for the Model Database (Example 1)

Notes:

1. F01KEY is the IMS key field (as defined in the IMS DBDGEN) of the root segment S01. This field is defined to DataRefresher with a SEQFLD=R keyword as the IMS key field.
2. Because of specific requirements of the application used in this example, the IMS key field F01KEY should not be propagated to one column. Instead it should be propagated as two key subfields (F01K01 and F01K02) to two DB2 columns. Therefore, the CREATE DXTPSB statement provides FIELD statements for F01K01 and F01K02. These two fields overlay the IMS key field F01KEY.
3. The segment has a variable-length and is therefore defined with the FORMAT=V keyword. It is the maximum segment length defined on the BYTES=126 keyword.
4. F02KEY is the IMS key field (as defined in the IMS DBDGEN) of the segment S02. This field is defined to DataRefresher with a SEQFLD=R keyword as the IMS key field.
5. Because of specific requirements of the application used in this example, the IMS key field F02KEY should not be propagated to one column. Instead it should be propagated as two key subfields (F02K01 and F02K02) to two DB2 columns. Therefore, the CREATE DXTPSB statement provides FIELD statements for F02K01 and F02K02. These two fields overlay the IMS key field F02KEY.
6. Field F02D07 is the length field of VG field F02D08. It must be located within the IMS segment before the VG field. F02D07 must contain the length of F02D08 expressed in bytes (*not* expressed in DBCS characters). As in most

real-life cases, the length field F02D07 will not be propagated to a DB2 column; therefore, the EXTRACT statement will not refer to F02D07.

7. Field F02D08 is a variable-length field. IMS DPROP requires that you provide an LFIELD keyword for a variable-length field. LFIELD=F02D07 indicates that F02D07 is the length field.

Defining the DXTVIEW: 1

DataRefresher requires a view description for each hierarchical path of the database. The DataRefresher command used to define the view for our example database is shown in Figure 455.

```
                                                NOTE

CREATE DXTVIEW NAME     = VIEW02,
               DXTPSB   = PSB00,                 1
               DXTPCB   = PCB001,                2
               SEGMENT  = S02,                   3
               MINSEGM  = S02,                   3
               FIELDS   = *  ;                   4
```

Figure 455. VIEW02 for Segments S01 and S02 (Example 1)

Notes:

1. The DXTPSB=PSB00 keyword identifies the previously created DXTPSB.
2. The DXTPCB=PCB001 keyword identifies the DXTPCB of the DXTPSB that describes the propagated IMS database.
3. For DXTVIEWs used with propagation requests: belonging to mapping case 1, both SEGMENT=S02 and MINSEGM=S02 identify the lowest segment of the hierarchical path of the database involved in the extract and propagation.
4. FIELDS=* tells DataRefresher that all fields of the described hierarchical path should be included in the DXTVIEW.

Providing DataRefresher UIM SUBMIT Commands and EXTRACT Statements: 1

Figure 456 on page 610 illustrates the DataRefresher SUBMIT command and EXTRACT statement used to define the propagation request that will propagate segment S02.

The DataRefresher SUBMIT/EXTRACT creates the propagation request into the IMS DPROP directory but does not load the data to the DB2 tables. This is the responsibility of the DataRefresher DEM. Refer to “Providing DataRefresher DEM control statements: 1” on page 610 for examples of a DataRefresher DEM control statement.

```
                                                          NOTE
SUBMIT   EXTID=PR02,
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,MAPDIR=HR,MAPCASE=1, ACTION=REPL,
                         ERROPT=BACKOUT',
               FORMAT=SOURCE,                                          1
               CD=JCS,                                                 2
               JCS=DDJCS01                                             3

EXTRACT
    INTO T02     (C01K01 NOT NULL,                                     4
                  C01K02 NOT NULL,
                  C02K01 NOT NULL,
                  C02K02 NOT NULL,
                  C02D03 NOT NULL WITH DEFAULT,
                  C02D04 NOT NULL WITH DEFAULT,
                  C02D05 NOT NULL WITH DEFAULT,
                  C02D06 NOT NULL WITH DEFAULT,
                  C02D08 NOT NULL WITH DEFAULT
                 )

    OPTIONS      (FLDERR(HALT))

    SELECT        F01K01,
                  F01K02,
                  F02K01,
                  F02K02,
                  F02D03,
                  F02D04,
                  F02D05,
                  F02D06,
                  F02D08                                              5
                 FROM VIEW02;                                         6
```

Figure 456. DataRefresher SUBMIT/EXTRACT for PR02 (Example 1)

Notes:

1. As recommended by IMS DPROP, FORMAT=SOURCE is specified, to reduce overhead during the extract.
2. CD=JCS is specified, which tells DataRefresher to create a control deck for the DB2 Load utility. DataRefresher DEM creates the control deck as a SYSIN file.
3. JCS=DDJCS01 is specified, which tells DataRefresher that you are providing JCL statements for the DB2 Load utility in the file allocated through a //DDJCS01 DD statement.
4. The INTO clause of the EXTRACT statement provides the same NOT NULL/NOT NULL WITH DEFAULT specifications as the DB2 CREATE TABLE statement. This ensures that the mapping performed by DataRefresher during the extract is compatible to the mapping performed by IMS DPROP during propagation.
5. F02D08, as defined in the DXTPSB, is a variable-length field. Its length field, F02D07, is not propagated and is therefore not included in the SELECT clause.
6. VIEW02 is the previously created DXTVIEW.

Providing DataRefresher DEM control statements: 1

Figure 457 on page 611 shows an example of a DataRefresher DEM control statement that you can use to perform the DataRefresher extract for PR02.

```
INITDEM   NAME=DEM01
USE DXTPSB=PSB00;
USE EXTID=PR02;
```

Figure 457. DataRefresher DEM control statements (Example 1)

Defining Propagation Requests with the MVG Input Tables ISPF Application: 1

Call the MVGIN Front End Application as described here to display the panel shown in Figure 458.

Starting the MVGIN Frontend Application Directly

To start the MVGIN Frontend Application directly, first allocate *prefix*.SEKYCLIB.EKYML00X to your //SYSEXEC or //SYSPROC. Then, enter the following command in an active ISPF/PDF environment:

```
EXEC 'prefix.SEKYCLIB(EKYML00X)'
```

You cannot pass a parameter to the procedure. However, when directly calling the MVGIN Frontend Application, you must define a high-level qualifier for the DPROP ISPF libraries in the REXX procedure EKYML00X. Edit the REXX procedure EKYML00X, specifying the high-level qualifier in the variable name EKYMHLQ.

Each time you call the MVGIN Frontend Application, you must provide the required parameters of the DPROP system for which you want to generate MVGIN jobs. These parameters are not stored for subsequent direct calls of the MVGIN Frontend Application.

```
EKYMP00E ------------- DPROP MVG Input Tables Application -------------------
Command ===>

                                                          DPROP Environ-
 1 - Insert, Update, Show, Delete a Propagation Request   ment  : (name)
 2 - Propagation Request Overview to Update, Show,        DPROP-
     Delete a Propagation Request                         System: (name)
 3 - Setup IMS and DPROP System Environment               DPROP-System-
 4 - Edit existing MVG Input Tables Insert/Delete Jobs    Number: (number)

 E - Return


   Enter the fully qualified name of the partitioned data set to be
   used to store the generated MVGIN job control:
      Data Set Name ===>
   Enter the volume serial name to be used to allocate a new data set:
      Volume Serial ===>


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 458. EKYMP00E: IMS DPROP MVG Input Tables Application (Example 1)

The first time you call the MVGIN Front End Application, you have to complete the setup panels and enter a data set name in the Data Set Name field.

After completing the setup, select option 1 to insert a new propagation request. The panel shown in Figure 459 is displayed.

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                        DPROP-System: (name)



     Option   ===> I       ( I - Insert a new PR                       )
                           ( U - Update an existing PR                 )
                           ( S - Show an existing PR                   )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR02


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                 ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Figure 459. EKYMP10E: Maintain a Propagation Request (Example 1 - PR02)

Type I in the Option field and PR02 in the propagation request Name field, then press Enter. After the application validates your input, the panel shown in Figure 460 is displayed.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR02

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 460. EKYMP15E: Propagation Request Definition Selection (Example 1 - PR02)

Select option 1 to enter the general propagation request specifications. The application displays the panel shown in Figure 461 on page 613.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------
Command ===>
                                                Propagation Request: PR02

                                                                    More:   - +
  PR Type                   ===> E         ( E, L, F or U                   )
  Mapping Case              ===> 1         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===>           ( ID, DENORM or blank            )
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                       <
    >                                                                       <
    >                                                                       <
    >                                                <
  User ID                   ===> T096606   ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                       <
    >                                                                       <
    >                                                                       <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 461. EKYMP20E: Propagation Request (Example 1 - PR02)

Type END on the command line and press Enter (or press F3) to return to panel EKYMP15E (shown in Figure 460 on page 612) with the state of option 1 changed to processed.

Select option 2 on panel EKYMP15E to define the DB2 table. The panel shown in Figure 462 on page 614 is displayed.

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                  Propagation Request: PR02

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===> T096606     ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> T02                  ( a name up to 18 bytes )


         Note: The DB2 table you enter is not checked for existence.





Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 462. EKYMP30E: DB2 Table (Example 1 - PR02)

Type END on the command line and press Enter (or press F3) to return to panel EKYMP15E (shown in Figure 460 on page 612) with the state of options 1 and 2 changed to `processed`.

Select option 3 on panel EKYMP15E to define the segments, fields and columns. The panel shown in Figure 463 is displayed:

```
EKYMP40E --- DPROP MVGIN - Entity Segment for a Mapping Case 1 PR -------------
Command ===>

    F - Fields for the entity segment            Propagation Request: PR02
    P - Fields for the parent segments           Database-
                                                 Organization: HDAM/OSAM

  IMS Database Name         ===> D00       ( a name up to 8 bytes           )
  Segment Name              ===> S02       ( a name up to 8 bytes           )


  PCB Label Name            ===>           ( a name up to 8 bytes           )


  Segment Exit Routine      ===>           ( a name up to 8 bytes, optional )
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760             )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                         )


Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 463. EKYMP40E: Entity Segment for a Mapping Case 1 Propagation Request (Example 1 - PR02)

To define the fields for the parent segments, type P on the command line and press Enter. The panel shown in Figure 464 on page 615 is displayed.

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------    NOTE
Command ===>                                                   SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR02
        sequence                             Entity/Internal Segment: S02

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S01         Segment Level:
FLDNAME  ===> F01K01                              POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          1
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C01K01
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S01         Segment Level:
FLDNAME  ===> F01K02                              POSITION ===> 3
BYTES    ===> 3         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          1
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C01K02
```

Figure 464. EKYMPPFE: Parent Segments Fields/Columns (Example 1 - PR02)

The key field F01KEY of segment S01 is not propagated to one column of the DB2 primary key. Instead, because of specific requirements of the application, it is propagated as two key subfields (F01K01, F01K02) to two columns of the DB2 primary key. Therefore, one field statement is specified for each of these two key subfields.

Note: The field statements describe the mapping of the two key subfields of segment S01. They map the fully concatenated key of the entity segment to the columns of the DB2 primary key.

Type END on the command line and press Enter (or press F3) to return to panel EKYMP40E.

To define the fields for the entity segment, type F on the command line of panel EKYMP40E and press Enter. The panel shown in Figure 465 on page 616 is displayed.

```
EKYMP41E -------- DPROP MVGIN - Entity Segment Fields/Columns -----------------    NOTE
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR02
    S - Sort in POSITION sequence                Entity Segment:      S02

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
FLDNAME  ===> F02K01                             POSITION ===> 3
BYTES    ===> 2         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          1
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02K01
-------------    2 ------------------------------------------------------------
FLDNAME  ===> F02K02                             POSITION ===> 5
BYTES    ===> 4         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          1
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02K02
-------------    3 ------------------------------------------------------------
FLDNAME  ===> F02D03                             POSITION ===> 9
BYTES    ===> 1         DATATYPE ===> B          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02D03
-------------    4 ------------------------------------------------------------
FLDNAME  ===> F02D04                             POSITION ===> 10
BYTES    ===> 2         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          2
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02D04
-------------    5 ------------------------------------------------------------
FLDNAME  ===> F02D05                             POSITION ===> 12
BYTES    ===> 3         DATATYPE ===> Z          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02D05
```

Figure 465. EKYMP41E: Entity Segment Fields/Columns (Example 1 - PR02) (Part 1 of 2)

The key field F02KEY of segment S02 is not propagated to one column of the DB2 primary key. Instead, because of specific requirements of the application, it is propagated as two key subfields (F02K01, F02K02) to two columns of the DB2 primary key. Therefore, one field statement is specified for each one of these two key subfields.

Notes:

1. The first two field statements describe the mapping of the two key subfields of segment S02. They map the fully concatenated key of the entity segment to the columns of the DB2 primary key.
2. This field statement describes the mapping of IMS packed field F02D04 to the DB2 decimal column C02D04. The value specified in the BYTES field, 2, describes the length of the field in bytes (while the length of the C02D04 column must be defined in the DB2 CREATE TABLE statement in digits).

Press F8 to display the panel shown in Figure 466 on page 617.

```
EKYMP41E -------- DPROP MVGIN - Entity Segment Fields/Columns -----------------    NOTE
Command ===>                                                   SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR02
    S - Sort in POSITION sequence                Entity Segment:      S02

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    6 ------------------------------------------------------------
FLDNAME  ===> F02D06                             POSITION ===> 15
BYTES    ===> 10        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02D06
-------------    7 ------------------------------------------------------------
FLDNAME  ===> F02D07                             POSITION ===> 25
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          1
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    8 ------------------------------------------------------------
FLDNAME  ===> F02D08                             POSITION ===> 27
BYTES    ===> 100       DATATYPE ===> VG         DATATYP2 ===>
SCALE    ===>           LENFIELD ===> F02D07                                   2,3
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02D08
```

Figure 466. EKYMP41E: Entity Segment Fields/Columns (Example 1 - PR02) (Part 2 of 2)

Notes:

1. This field statement provides the description of the length field F02D07 of the variable-length field F02D08. In this example, as in most real-life cases, the length-field is not propagated and has no target column. Therefore, the COLNAME field contains blank.
2. The field statement for field F02D08 provides the description of a variable-length field. Therefore, as required by IMS DPROP, this statement specifies the name of the length field in the LENFIELD field.
3. The field statement for field F02D08 describes a variable-length graphic field. The value specified in the BYTES field , 100, describes the maximum length of the field in bytes (while the length of the C02D08 column must be defined in the DB2 CREATE TABLE statement in DBCS characters). Length field F02D07 must contain the length, in bytes, of F02D08.

Type END on the command line and press Enter (or press F3) to return to the panel shown in Figure 463 on page 614. Repeat the procedure to return to panel EKYMP15E (shown in Figure 460 on page 612) with the state of options 1, 2, and 3 changed to `processed`.

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. The panel shown in Figure 467 on page 618 is overlaid on panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

    ______DPROP MVGIN - Generate MVG Input Tables Insert Job_______  (name)
   | EKYMPIIE ==>                                                   | 02
   |                                                                |
   |                                  DPROP-System:        (name)   | s
 1 |                                  Propagation Request: PR02     | d *
 2 |                                                                | d *
 3 |  Enter the member name to be used to store the generated       | d *
   |  MVG input tables insert job.  The job is stored in data set   | d *
 4 |  "(library)".                                                  | d *
   |                                                                |
 5 |     Member Name    ===> PR02                                   |
   |                                                                |
 S | Press <ENTER> to continue or enter END command to return.      |
   |                                                                |
 E |                                                                |
   |________________________________________________________________|


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 467. EKYMP15E: Propagation Request Definition Selection (Example 1 - PR02)

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Providing MVGU control statements: 1

Figure 468 describes the IMS DPROP MVGU control statement that tells the MVGU to create PR02 in the IMS DPROP directory based on data stored in the MVG input tables.

```
CREATE PR=PR02
```

Figure 468. MVGU control statements to Create the PR

Defining Propagation Requests in the MVG Input Tables with SQL Insert Statements: 1

Figure 469 on page 619 describes SPUFI statements that define propagation requests in the MVG input tables.

Typically, propagation requests are described in the MVG input tables by a program you have provided that has a dictionary as input. It is unusual to describe propagation requests in the MVG input tables with SPUFI statements.

As shown in Figure 469 on page 619, the following rows are inserted into the MVG input tables:

- One propagation request row containing general information about PR02
- One SEG row
  - For the entity segment, S02
  - For the only physical parent/ancestor segment, S01
- One TAB row identifying the target table, T02
- One FLD row
  - For each propagated field
  - For the length-field of the variable-length field, F02D08

The two key fields (F01KEY and F01KEY) of segments S01 and S02 are not propagated to two columns of the DB2 primary key. Instead, because of specific requirements of the application, they are propagated as four key subfields (F01K01, F01K02, F02K01, F02K02) to four columns of the DB2 primary key. Therefore, one row is inserted into the FLD table for each one of these four key subfields.

```
INSERT INTO T096606.DPRIPR
            ( PRID,     USERID,   PRTYPE, MAPCASE, MAPDIR,
              ERROPT,   ACTION)
      VALUES ('PR02',   'T096606', 'E',   '1',      'HR',
             'BACKOUT','REPL')                            ;

INSERT INTO T096606.DPRISEG
            ( PRID,   DBNAME, SEGNAME, ROLE )
      VALUES ('PR02', 'D00',  'S01',   'P' )              ;

INSERT INTO T096606.DPRISEG
            ( PRID,   DBNAME, SEGNAME, ROLE )
      VALUES ('PR02', 'D00',  'S02',   'E' )              ;

INSERT INTO T096606.DPRITAB
            ( PRID,   TABQUAL,   TABNAME )
      VALUES ('PR02', 'T096606', 'T02')                     ;

INSERT INTO T096606.DPRIFLD                                          1
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02',   'D00',    'S01',   'F01K01' ,
             'T096606','T02',    'C01K01',
             'C ',      1,        2)                       ;

INSERT INTO T096606.DPRIFLD                                          1
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02',   'D00',    'S01',   'F01K02' ,
             'T096606','T02',    'C01K02',
             'C ',      3,        3)                       ;

INSERT INTO T096606.DPRIFLD                                          1
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02',   'D00',    'S02',   'F02K01' ,
             'T096606','T02',    'C02K01',
             'C ',      3,        2)                       ;

INSERT INTO T096606.DPRIFLD                                          1
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02',   'D00',    'S02',   'F02K02' ,
             'T096606','T02',    'C02K02',
             'C ',      5,        4)                       ;
```

Figure 469. SPUFI Statements to Define PR02 in the MVG Input Tables (Part 1 of 2)

```
                                                                        NOTE
INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02',   'D00',    'S02',   'F02D03' ,
              'T096606','T02',    'C02D03',
              'B ',      9,        1)                      ;

INSERT INTO T096606.DPRIFLD                                          2
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02',   'D00',    'S02',   'F02D04' ,
              'T096606','T02',    'C02D04',
              'P ',      10,       2)                      ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02',   'D00',    'S02',   'F02D05' ,
              'T096606','T02',    'C02D05',
              'Z ',      12,       3)                      ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02',   'D00',    'S02',   'F02D06' ,
              'T096606','T02',    'C02D06',
              'C ',      15,       10)                     ;

INSERT INTO T096606.DPRIFLD                                          3
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02',   'D00',    'S02',   'F02D07' ,
              'T096606','T02',    '      ',
              'H ',      25,       2 )                     ;

INSERT INTO T096606.DPRIFLD                                          4,5
            ( PRID,     DBNAME,   SEGNAME, FLDNAME, LENFIELD ,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02',   'D00',    'S02',   'F02D08' , 'F02D07',
              'T096606','T02',    'C02D08',
              'VG',      27,       100)                    ;
```

Figure 469. SPUFI Statements to Define PR02 in the MVG Input Tables (Part 2 of 2)

Notes:

1. The first four INSERT INTO T096606.DPRIFLD statements describe the mapping for the four key subfields. They map the fully concatenated key of the entity segment to the columns of the DB2 primary key. The first two of these statements insert S01 into the SEGNAME column (because F01K01 and F01K02 are fields of S01, not S02).
2. The INSERT INTO T096606.DPRIFLD statement describes the mapping of IMS packed field F02D04 to the DB2 decimal column C02D04. The value inserted into the BYTES column describes the length of the field in bytes (while the length of the C02D04 column must be defined in the DB2 CREATE TABLE statement in digits).

3. The INSERT INTO T096606.DPRIFLD statement for field F02D07 provides the description of the length field of variable-length field F02D08. In this example, as in most real-life cases, the length-field is not propagated and has no target column. Therefore, this statement inserts a blank into the COLNAME column.
4. The INSERT INTO T096606.DPRIFLD statement for field F02D08 provides the description of a variable-length field. Therefore, as required by IMS DPROP, this statement inserts the name of the length field into the LENFIELD column.
5. The INSERT INTO T096606.DPRIFLD statement for field F02D08 describes a variable-length graphic field. The value inserted into the BYTES column describes the maximum length of the field in bytes (while the length of the C02D08 column must be defined in the DB2 CREATE TABLE statement in DBCS characters). Length field F02D07 must contain the length, in bytes, of F02D08.

Providing MVGU control statements: 1

Figure 470 describes the IMS DPROP MVGU control statement that tells the MVGU to create PR02 in the IMS DPROP directory based on data stored in the MVG input tables.

```
CREATE PR=PR02
```

Figure 470. MVGU control statements to Create the PR

Example 2: Propagation with Mapping Case 2

In this example, the entity segment S13 and the extension segments S14 and S15 of the database model shown in Figure 471 are propagated. The propagation is done with a propagation request belonging to mapping case 2.

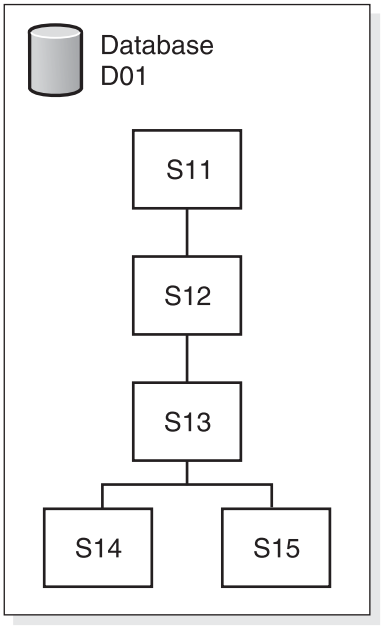

Figure 471. Database Model for Example 2

This example describes the definitions to propagate:

- The fields of the fully concatenated key of the entity segment S13
- Data fields stored in the entity segment S13
- Data fields stored in the extension segments S14 and S15.

Overview of the Propagation Performed: 2

Figure 472 illustrates the propagation of the entity segment S13 and the extension segments S14 and S15, by propagation request PR3, to table T13.

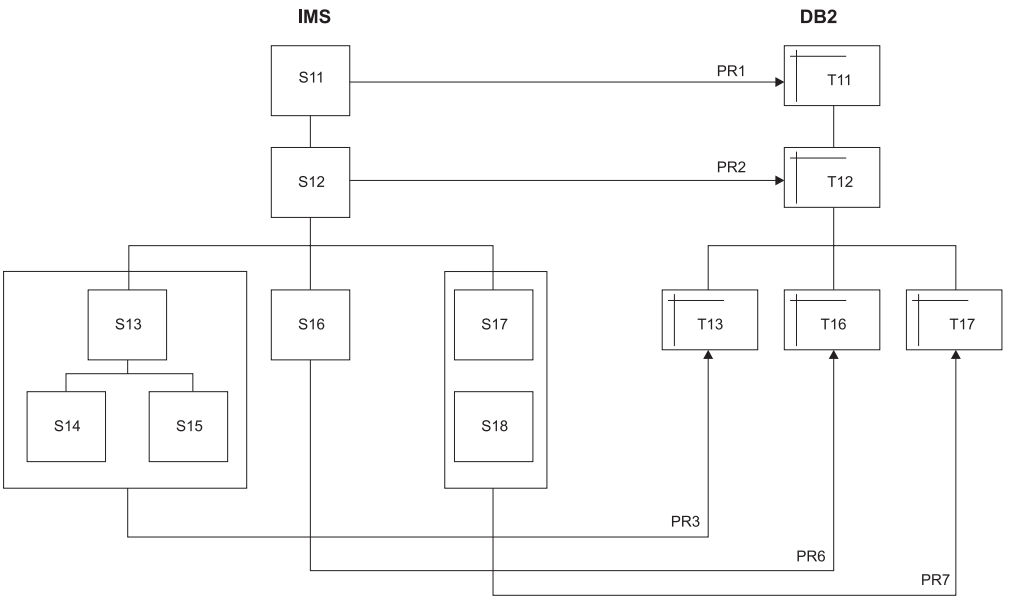

Figure 472. Overview of Propagation of Database D01

Figure 473 on page 623 illustrates the mapping between the fully concatenated key of the entity segment S13 and the DB2 primary key of table T13.

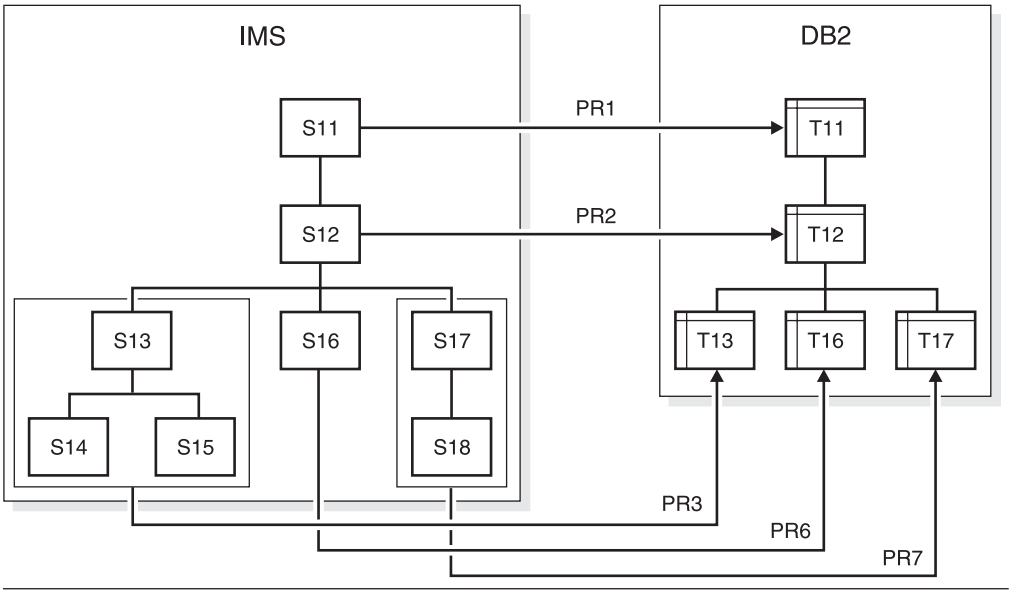

Figure 473. Key Mapping Between the Entity Segment S13 and T13. Mapping of the fully concatenated key fields of the entity segment to the DB2 primary key columns

Notes:

1. The fully concatenated key of the entity segment S13 is propagated, as IMS DPROP recommends, to the DB2 primary key.

   The fully concatenated key of S13 consists of:
   - The key field of S11 (which has been subdivided into the subfields F11K01 and F11K02)
   - The key field of S12 (F12K02)
   - The key field of S13 (which has been subdivided into the subfields F13K02 and F13K03)
2. The fully concatenated key of S13 is *not* mapped by propagating the 3 key fields F11KEY, F12KEY, and F13KEY to 3 columns of the DB2 primary key. Instead, the fully concatenated key is propagated to the DB2 primary key by mapping subfields of F11KEY and F13KEY to columns of the DB2 primary key:
   - The subfield of F11KEY, F11K01, is mapped to C11K01
   - The subfield of F11KEY, F11K02, is mapped to C11K02
   - The key field F12K02 is mapped to C12K02
   - The subfield of F13KEY, F13K02, is mapped to C13K02
   - The subfield of F13KEY, F13K03, is mapped to C13K03

Field Mapping Overview: 2

Table 48 on page 624 shows the characteristics of the source IMS fields and the target DB2 columns. The table contains the following information:

- For the IMS source field:
  - The name of the IMS field (as defined to IMS DPROP either on the NAME keyword of the DataRefresher FIELD statement, or in the FLDNAME column of the DPRIFLD MVG input table)
  - The data type (as defined to IMS DPROP either on the TYPE keyword of the DataRefresher FIELD statement, or in the DATATYPE column of the DPRIFLD MVG input table)

  - The length in bytes (as defined to IMS DPROP either on the BYTES keyword of the DataRefresher FIELD statement or in the BYTES column of the DPRIFLD MVG input table)
- For the DB2 target column:
  - The name of the target DB2 column (as defined to DB2 in the CREATE TABLE statement)
  - The data type of the target DB2 column (as defined to DB2 in the CREATE TABLE statement)

Table 48. Source Segments: S13 (and its concatenated key), S14 and S15 - Target Table: T13

Source IMS fields				Target DB2 columns	
Field Name	**See Note Below**	**Data Type**	**Length (in Bytes)**	**Column Name**	**Data Type**
F11K01	1	C	2	C11K01	CHAR(2)
F11K02	1	P	4	C11K02	DECIMAL(7)
F12K02	1	P	4	C12K02	DECIMAL(7)
F13D01	3	H	2	C13D01	SMALLINT
F13K02	2	P	3	C13K02	DECIMAL(5)
F13K03	2	P	5	C13K03	DECIMAL(9)
F13D04	3	H	2	C13D04	SMALLINT
F13D05	3	F	4	C13D05	INTEGER
F13D06	3	H	2	C13D06	SMALLINT
F14D01	3	C	2	C14D01	CHAR(2)
F14D02	3	C	5	C14D02	CHAR(5)
F14D03	3	C	8	C14D03	CHAR(8)
F15D01	3	P	2	C15D01	DECIMAL(3)
F15D02	3	C	30	C15D02	CHAR(30)

Notes:

1. F11K01, F11K02 and F12K02 are located in the concatenated key of the entity segment. They are the key fields (or subfields of the key fields) of the physical parent segment S12 and of the physical ancestor segment S11. They are propagated to columns of the DB2 primary key of T13.
2. F13K02 and F13K03 are subfields of the key field of entity segment S13. They are propagated to columns of the DB2 primary key of T13.
3. The other fields are non-key fields of the entity segment S13 and of the extension segments S14 and S15.

DBD Definitions: 2

The DBD control statements for the model database are shown in Figure 474 on page 625.

```
  DBD     NAME=D01,ACCESS=HISAM,EXIT=(EKYRUP00,CASCADE),VERSION=128
  DATASET DD1=D01DDN1,OVFLW=D01DDN2,DEVICE=3380
*
  SEGM    NAME=S11,BYTES=24,PARENT=0
  FIELD   NAME=(F11KEY,SEQ,U),TYPE=C,BYTES=6,START=1
  FIELD   NAME=F11K01,TYPE=C,BYTES=2,START=1
  FIELD   NAME=F11K02,TYPE=P,BYTES=4,START=3
*
  SEGM    NAME=S12,BYTES=26,PARENT=S11
  FIELD   NAME=(F12KEY,SEQ,U),TYPE=C,BYTES=4,START=2
  FIELD   NAME=F12K02,TYPE=P,BYTES=4,START=2
*
  SEGM    NAME=S13,BYTES=18,PARENT=S12
  FIELD   NAME=(F13KEY,SEQ,U),TYPE=C,BYTES=8,START=3
  FIELD   NAME=F13D01,TYPE=X,BYTES=2,START=1
  FIELD   NAME=F13K02,TYPE=P,BYTES=3,START=3
  FIELD   NAME=F13K03,TYPE=P,BYTES=5,START=6
  FIELD   NAME=F13D04,TYPE=X,BYTES=2,START=11
  FIELD   NAME=F13D05,TYPE=X,BYTES=4,START=13
  FIELD   NAME=F13D06,TYPE=X,BYTES=2,START=17
*
  SEGM    NAME=S14,BYTES=15,PARENT=S13
  FIELD   NAME=F14D01,TYPE=C,BYTES=2,START=1
  FIELD   NAME=F14D02,TYPE=C,BYTES=5,START=3
  FIELD   NAME=F14D03,TYPE=C,BYTES=8,START=8
*
  SEGM    NAME=S15,BYTES=32,PARENT=S13
  FIELD   NAME=F15D01,TYPE=P,BYTES=2,START=1
  FIELD   NAME=F15D02,TYPE=C,BYTES=30,START=3
*
  DBDGEN
  FINISH
  END
```

Figure 474. DBD control statements for the Model Database

The EXIT keyword in the DBD control statement tells IMS to give control to EKYRUP00, the IMS DPROP-provided DL/I Data Capture Exit routine, when a DL/I call updates a segment of the database D01.

You can see that in the DataRefresher EXTRACT statement or the SPUFI statements used to define the propagation request in the MVG input tables, the fully concatenated key of S13 is not propagated by mapping the key fields F11KEY, F12KEY, and F13KEY. Instead, the fully concatenated key is propagated (to the DB2 primary key) by mapping subfields of F11KEY and F13KEY.

Usually, IMS DPROP and DataRefresher do not require you to define the non-key fields in the IMS DBD. In this example, the non-key fields are not defined in the IMS DBD.

Table Definitions: 2

Figure 475 on page 626 describes the SQL statements to create the target table T13. As required by IMS DPROP, table T13 has a DB2 primary key. And, as recommended by IMS DPROP, the DB2 primary key will be the propagated fully concatenated key of the entity segment.

```
CREATE TABLE T096606.T13
          (C11K01   CHAR(2)       NOT NULL,
           C11K02   DECIMAL(7)    NOT NULL,
           C12K02   DECIMAL(7)    NOT NULL,
           C13D01   SMALLINT      NOT NULL WITH DEFAULT,
           C13K02   DECIMAL(5)    NOT NULL,
           C13K03   DECIMAL(9)    NOT NULL,
           C13D04   SMALLINT      NOT NULL WITH DEFAULT,
           C13D05   INTEGER       NOT NULL WITH DEFAULT,
           C13D06   SMALLINT      NOT NULL WITH DEFAULT,
           C14D01   CHAR(2)       NOT NULL WITH DEFAULT,
           C14D02   CHAR(5)       NOT NULL WITH DEFAULT,
           C14D03   CHAR(8)       NOT NULL WITH DEFAULT,
           C15D01   DECIMAL(3)    NOT NULL WITH DEFAULT,
           C15D02   CHAR(30)      NOT NULL WITH DEFAULT,
        PRIMARY KEY (C11K01,C11K02,C12K02,C13K02,C13K03))
        IN DU096606.ST13;

CREATE UNIQUE INDEX T096606.TX13
       ON T096606.T13 (C11K01,C11K02,C12K02,C13K02,C13K03)
       USING VCAT KOE ;
```

Figure 475. Target Table T13

Defining Propagation Requests with DataRefresher: 2

Defining the DXTPSB: 2

DataRefresher requires that you describe the IMS database to DataRefresher by providing a CREATE DXTPSB command. The DataRefresher control statements for the model database are shown in Figure 476 on page 627. The control statements, which are highlighted numerically (**1** or **2**, for example), are further described below the figure.

```
                                                                  NOTE
CREATE  DXTPSB   NAME=PSB1

         DXTPCB   NAME=PSB1PCB1,DBACCESS=HISAM,DBNAME=D01

        SEGMENT   NAME=S11,PARENT=0,BYTES=24
          FIELD   NAME=F11KEY,TYPE=C,BYTES=6,START=1,SEQFLD=R        1
          FIELD   NAME=F11K01,TYPE=C,BYTES=2,START=1                 2
          FIELD   NAME=F11K02,TYPE=P,BYTES=4,START=3                 2

        SEGMENT   NAME=S12,PARENT=S11,BYTES=26
          FIELD   NAME=F12K02,TYPE=P,BYTES=4,START=2                 3

        SEGMENT   NAME=S13,PARENT=S12,BYTES=18
          FIELD   NAME=F13D01,TYPE=H,BYTES=2,START=1
          FIELD   NAME=F13K02,TYPE=P,BYTES=3,START=3                 4
          FIELD   NAME=F13K03,TYPE=P,BYTES=5,START=6                 4
          FIELD   NAME=F13D04,TYPE=H,BYTES=2,START=11
          FIELD   NAME=F13D05,TYPE=F,BYTES=4,START=13
          FIELD   NAME=F13D06,TYPE=H,BYTES=2,START=17

        SEGMENT   NAME=S14,PARENT=S13,BYTES=15,MAXNBR=1
          FIELD   NAME=F14D01,TYPE=C,BYTES=2,START=1
          FIELD   NAME=F14D02,TYPE=C,BYTES=5,START=3
          FIELD   NAME=F14D03,TYPE=C,BYTES=8,START=8

        SEGMENT   NAME=S15,PARENT=S13,BYTES=32,MAXNBR=1
          FIELD   NAME=F15D01,TYPE=P,BYTES=2,START=1
          FIELD   NAME=F15D02,TYPE=C,BYTES=30,START=3 ;
```

Figure 476. DataRefresher control statements for the Model Database

Notes:

1. F11KEY is the IMS key field (as defined in the IMS DBDGEN) of the root segment S11. This field is defined to DataRefresher with a SEQFLD=R keyword as the IMS key field.
2. Because of specific requirements of the application used in this example, the IMS key field F11KEY should not be propagated to one column. Instead it should be propagated as two key subfields (F11K01 and F11K02) to two DB2 columns. Therefore, the CREATE DXTPSB statement provides FIELD statements for F11K01 and F11K02. These two fields overlay the IMS key field F11KEY.
3. The IMS key field F12K12 of the segment S12 is defined to DataRefresher. Since the field was not defined to DataRefresher as the IMS key field (it was not defined with the SEQFLD=R operand), the name defined to DataRefresher does not need to match the name defined to IMS in the DBD.
4. F13KEY is the IMS key field (as defined in the IMS DBDGEN) of the segment S13. This field is not defined to DataRefresher.

   Because of specific requirements of the application used in this example, the IMS key field F13KEY should not be propagated to one column. Instead it should be propagated as two key subfields (F13K02 and F13K03) to two DB2 columns. Therefore, the CREATE DXTPSB statement provides FIELD statements for F13K02 and F13K03. These two fields overlay the IMS key field F13KEY.

Defining the DXTVIEW: 2

DataRefresher requires a view description for each hierarchical path of the database.

Because the mapping case 2 propagation request described in this example has two extension segments, it involves two hierarchical paths of the database. Therefore, two DXTVIEWs must be defined to DataRefresher.

The DataRefresher commands used to define the views for our example database are shown in Figure 477.

```
                                                    NOTE
CREATE DXTVIEW   NAME=VIEWA,
                 DXTPSB=PSB1,                       1
                 DXTPCB=PSB1PCB1,                   2
                 SEGMENT=S14,MINSEGM=S13,           3
                 FIELDS=(F11K01,F11K02,             5
                         F12K02,
                         F13D01,F13K02,F13K03,F13D04,F13D05,F13D06,
                         F14D01,F14D02,F14D03);

CREATE DXTVIEW   NAME=VIEWB,
                 DXTPSB=PSB1,                       1
                 DXTPCB=PSB1PCB1,                   2
                 SEGMENT=S15,MINSEGM=S13,           4
                 FIELDS=(F11K01,F11K02,             5
                         F12K02,
                         F13D01,F13K02,F13K03,F13D04,F13D05,F13D06,
                         F15D01,F15D02);
```

Figure 477. VIEWA and VIEWB for the Two Extension Segments/Hierarchical Paths

Notes:

1. The DXTPSB=PSB1 keyword identifies the previously created DXTPSB.
2. The DXTPCB=PSB1PCB1 keyword identifies the DXTPCB of the DXTPSB that describes the propagated IMS database.
3. For DXTVIEWs used with propagation requests belonging to mapping case 2, SEGMENT=S14 identifies the extension segment; MINSEGM=S13 identifies the entity segment.
4. For DXTVIEWs used with propagation requests belonging to mapping case 2, SEGMENT=S15 identifies the extension segment; MINSEGM=S13 identifies the entity segment.
5. The FIELDS= operand identifies fields in the described hierarchical path that should be included in the DXTVIEW.

Providing DataRefresher UIM SUBMIT Commands and EXTRACT Statements: 2

Figure 478 on page 629 illustrates the DataRefresher SUBMIT command and EXTRACT statement used to define the propagation request that will propagate the entity segment S13 and the extension segments S14 and S15.

The DataRefresher SUBMIT/EXTRACT creates the PR in the IMS DPROP directory but does not load the data to the DB2 tables. This is the responsibility of the DataRefresher DEM. Refer to “Providing DataRefresher DEM control statements: 2 ” on page 630 for examples of a DataRefresher DEM control statement.

```
                                                                    NOTE

SUBMIT   EXTID=PR3,
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,
                         MAPCASE=2,
                         MAPDIR=HR,
                         ERROPT=BACKOUT,
                         ACTION=REPL',
               FORMAT=SOURCE,                                       1
               CD=JCS,                                              2
               JCS=DDJCS11                                          3

EXTRACT  INTO  T096606.T13 (C11K01 NOT NULL,                        4
                            C11K02 NOT NULL,
                            C12K02 NOT NULL,
                            C13D01 NOT NULL WITH DEFAULT,
                            C13K02 NOT NULL,
                            C13K03 NOT NULL,
                            C13D04 NOT NULL WITH DEFAULT,
                            C13D05 NOT NULL WITH DEFAULT,
                            C13D06 NOT NULL WITH DEFAULT,
                            C14D01 NOT NULL WITH DEFAULT,
                            C14D02 NOT NULL WITH DEFAULT,
                            C14D03 NOT NULL WITH DEFAULT,
                            C15D01 NOT NULL WITH DEFAULT,
                            C15D02 NOT NULL WITH DEFAULT)

         OPTIONS            (FLDERR(HALT))

         SELECT             VIEWA.F11K01,                           5
                            VIEWA.F11K02,
                            VIEWA.F12K02,
                            VIEWA.F13D01,
                            VIEWA.F13K02,
                            VIEWA.F13K03,
                            VIEWA.F13D04,
                            VIEWA.F13D05,
                            VIEWA.F13D06,
                            VIEWA.F14D01,
                            VIEWA.F14D02,
                            VIEWA.F14D03,
                            VIEWB.F15D01,                           6
                            VIEWB.F15D02

         FROM               VIEWA,                                  5
                            VIEWB;                                  6
```

Figure 478. DataRefresher SUBMIT/EXTRACT for PR3

Notes:

1. As recommended by IMS DPROP, FORMAT=SOURCE is specified to reduce overhead during the extract.
2. CD=JCS is specified, which tells DataRefresher to create a control deck for the DB2 Load utility. DataRefresher DEM creates the control deck as a SYSIN file.
3. JCS=DDJCS11 is specified, which tells DataRefresher that you are providing JCL statements for the DB2 Load utility in the file allocated through a //DDJCS11 DD statement.
4. The INTO clause of the EXTRACT statement provides the same NOT NULL/NOT NULL WITH DEFAULT specifications as the DB2 CREATE TABLE

statement. This ensures that the mapping performed by DataRefresher during the extract is compatible to the mapping performed by IMS DPROP during propagation.

5. VIEWA is the previously created DXTVIEW for one of the two extension segments/hierarchical paths.
6. VIEWB is the previously created DXTVIEW for the second extension segment/hierarchical path.

Providing DataRefresher DEM control statements: 2

Figure 479 shows an example of a DataRefresher DEM control statement that you can use to perform the DataRefresher extract for PR3.

```
INITDEM NAME=DEM1
USE DXTPSB=PSB1;
USE EXTID=PR3;
```

Figure 479. DataRefresher DEM control statement for an Extract

Defining Propagation Requests with the MVG Input Tables ISPF Application: 2

Call the MVGIN Front End Application as described here to display the panel shown in Figure 480 on page 631.

Starting the MVGIN Frontend Application Directly

To start the MVGIN Frontend Application directly, first allocate *prefix*.SEKYCLIB.EKYML00X to your //SYSEXEC or //SYSPROC. Then, enter the following command in an active ISPF/PDF environment:

```
EXEC 'prefix.SEKYCLIB(EKYML00X)'
```

You cannot pass a parameter to the procedure. However, when directly calling the MVGIN Frontend Application, you must define a high-level qualifier for the DPROP ISPF libraries in the REXX procedure EKYML00X. Edit the REXX procedure EKYML00X, specifying the high-level qualifier in the variable name EKYMHLQ.

Each time you call the MVGIN Frontend Application, you must provide the required parameters of the DPROP system for which you want to generate MVGIN jobs. These parameters are not stored for subsequent direct calls of the MVGIN Frontend Application.

```
EKYMP00E ------------- DPROP MVG Input Tables Application ---------------------
Command ===>

                                                              DPROP Environ-
   1 - Insert, Update, Show, Delete a Propagation Request     ment  : (name)
   2 - Propagation Request Overview to Update, Show,          DPROP-
       Delete a Propagation Request                           System: (name)
   3 - Setup IMS and DPROP System Environment                 DPROP-System-
   4 - Edit existing MVG Input Tables Insert/Delete Jobs      Number: (number)

   E - Return


     Enter the fully qualified name of the partitioned data set to be
     used to store the generated MVGIN job control:
        Data Set Name ===>
     Enter the volume serial name to be used to allocate a new data set:
        Volume Serial ===>


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 480. EKYMP00E: DPROP MVG Input Tables Application (Example 2)

The first time you call the MVGIN Front End Application, you have to complete the setup panels, type a data set name in the Data Set Name field, and press Enter.

After completing the setup, select option 1 to insert a new propagation request. The panel shown in Figure 481 is displayed.

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                     DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR3


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Figure 481. EKYMP10E: DPROP MVGIN - Maintain a Propagation Request (Example 2)

Type I in the Option field and PR3 in the propagation request Name field and press Enter. After your input is validated, the panel shown in Figure 482 on page 632 is displayed, with the state of all options shown as `not processed`.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR3

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 482. EKYMP15E: Propagation Request Definition Selection (Example 2 - PR3)

Select option 1 to enter the general propagation request specifications. The panel shown in Figure 483 is displayed.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------
Command ===>
                                                Propagation Request: PR3

                                                                      More:   - +
  PR Type                   ===> E         ( E, L, F or U                   )
  Mapping Case              ===> 2         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===>           ( ID, DENORM or blank            )
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <
  User ID                   ===>           ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 483. EKYMP20E: DPROP MVGIN - Propagation Request (Example 2 - PR3)

Type END on the command line and press Enter (or press F3) to return to panel EKYMP15E (shown in Figure 482 on page 632) with the state of option 1 shown as `processed`.

Select option 2 on panel EKYMP15E to define the DB2 table. The panel shown in Figure 484 is displayed:

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                   Propagation Request: PR3

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===>            ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> T13                  ( a name up to 18 bytes )


         Note: The DB2 table you enter is not checked for existence.





Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 484. EKYMP30E: DB2 Table (Example 3 - PR3)

Type END on the command line and press Enter (or press F3) to return to panel EKYMP15E (shown in Figure 482 on page 632) with the state of options 1 and 2 changed to `processed`.

Select option 3 on panel EKYMP15E to define the segments, fields, and columns. The panel shown in Figure 485 is displayed.

```
EKYMP50E - DPROP MVGIN - Segments and Fields for a Mapping Case 2 PR ----------
Command ===>


                                                 Propagation Request: PR3

                                        State
    1 - Entity Segment              not processed

    2 - Extension Segments and      not processed
        Fields/Columns              not processed


    E - Return





Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 485. EKYMP50E: Segments and Fields for a Mapping Case 2 Propagation Request (Example 2 - PR3)

Select option 1 to define the entity segment. The panel shown in Figure 486 is displayed.

```
EKYMP51E --- DPROP MVGIN - Entity Segment for a Mapping Case 2 PR -------------
Command ===>

                                                Propagation Request: PR3
                                                Database-
                                                Organization: HISAM/VSAM

  IMS Database Name         ===> D01       ( a name up to 8 bytes           )
  Segment Name              ===> S13       ( a name up to 8 bytes           )


  PCB Label Name            ===>           ( a name up to 8 bytes           )


  Segment Exit Routine      ===>           ( a name up to 8 bytes, optional )
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760             )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                         )


Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 486. EKYMP51E: Entity Segment for a Mapping Case 2 Propagation Request (Example 2 - PR3)

Type END on the command line and press Enter (or press F3) to return to panel EKYMP50E (shown in Figure 485 on page 633) with option 1 changed to `processed`.

Select option 2 to define the extension segments and the fields and columns. The panel shown in Figure 487 is displayed.

```
EKYMP52E ------------- DPROP MVGIN - Extension Segments ------------------------
Command ===>                                                  SCROLL ===> HALF

    F - Fields for the entity and ext. segments  Propagation Request: PR3
    P - Fields for the parent segments           Entity Segment:      S13

  - Enter an X in the X column below to define an extension segment
  - Remove the X from the X column to unselect an extension segment

Press <ENTER> to validate your input.  Enter END command to return.

X  Seg Name  Seg Exit  Length of Segment processed  Format of Segment processed
                       by the Segment Exit Routine  by the Segment Exit Routine
                       ( from 1 up to 32760 )       ( F or V )

------------------------------------------------------------------------------  X  S14
------------------------------------------------------------------------------  X  S15
------------------------------------------------------------------------------
****************************** BOTTOM OF DATA ******************************
```

Figure 487. EKYMP52E: Extension Segments (Example 2 - PR3)

To define the fields for the parent segments, type P on the command line and press Enter. The panel shown in Figure 488 on page 635 is displayed:

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------    NOTE
Command ===>                                                   SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR3
        sequence                             Entity/Internal Segment: S13

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11K01                             POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          2
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C11K01
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11K02                             POSITION ===> 3
BYTES    ===> 4         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          1,2
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C11K02
-------------    3 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12K02                             POSITION ===> 2
BYTES    ===> 4         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          1,2
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C12K02
```

Figure 488. EKYMPPFE: Parent Segments Fields/Columns (Example 2 - PR3)

The key field F11KEY of segment S11 is not propagated to one column of the DB2 primary key. Instead, because of specific requirements of the application, it is propagated as two key subfields (F11K01, F11K02) to two columns of the DB2 primary key. Therefore, one field statement is specified for each one of these two key subfields.

Notes:

1. These field statements describe the mapping of IMS packed fields F11K02 and F12K02 to the DB2 target columns. The values specified in the BYTES fields describe the length of the field in bytes (while the length of the column must be defined in the DB2 CREATE TABLE statement in digits).
2. These field statements describe the mapping of the two key subfields of segment S11 and of the one key field of segment S12. They map the fully concatenated key of the entity segment to the columns of the DB2 primary key.

Type END on the command line and press Enter (or press F3) to return to panel EKYMP52E.

To define the fields for the entity segment, type F on the command line of panel EKYMP52E and press Enter. The panel shown in Figure 489 on page 636 is displayed:

```
EKYMP53E  DPROP MVGIN - Entity and Extension Segments Fields/Columns ----------    NOTE
Command ===>                                                   SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR3
    S - Sort in SEGNAME, POSITION sequence       Entity Segment:      S13

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13K02                                POSITION ===> 3
BYTES    ===> 3         DATATYPE ===> P             DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          1,2
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13K02
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13K03                                POSITION ===> 6
BYTES    ===> 5         DATATYPE ===> P             DATATYP2 ===>
SCALE    ===>           LENFIELD ===>                                          1,2
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13K03
-------------    3 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13D01                                POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> H             DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D01
```

Figure 489. EKYMP53E: Entity and Extension Segments Fields/Columns (Example 2 - PR3) (Part 1 of 3)

The key field F13KEY of segment S13 is not propagated to one column of the DB2 primary key. Instead, because of specific requirements of the application, it is propagated as two key subfields (F13K02, F13K03) to two columns of the DB2 primary key. Therefore, one field statement is specified for each one of these two key subfields.

Notes:

1. The first two field statements describe the mapping of the two key subfields of segment S13. They map the fully concatenated key of the entity segment to the columns of the DB2 primary key.
2. This field statement describes the mapping of IMS packed fields F13K02 and F13K03 to the DB2 target columns. The value specified in the BYTES fields describes the length of the field in bytes (while the length of the column must be defined in the DB2 CREATE TABLE statement in digits).

```
EKYMP53E  DPROP MVGIN - Entity and Extension Segments Fields/Columns ----------
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR3
    S - Sort in SEGNAME, POSITION sequence       Entity Segment:      S13

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    4 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13D04                             POSITION ===> 11
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D04
-------------    5 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13D05                             POSITION ===> 13
BYTES    ===> 4         DATATYPE ===> F          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D05
-------------    6 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13D06                             POSITION ===> 17
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D06
-------------    7 ------------------------------------------------------------
SEGNAME  ===> S14
FLDNAME  ===> F14D01                             POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C14D01
-------------    8 ------------------------------------------------------------
SEGNAME  ===> S14
FLDNAME  ===> F14D02                             POSITION ===> 3
BYTES    ===> 5         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C14D02
```

Figure 490. EKYMP53E: Entity and Extension Segments Fields/Columns (Example 2 - PR3) (Part 2 of 3)

```
EKYMP53E  DPROP MVGIN - Entity and Extension Segments Fields/Columns ----------
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR3
    S - Sort in SEGNAME, POSITION sequence       Entity Segment:      S13

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    9 ------------------------------------------------------------
SEGNAME  ===> S14
FLDNAME  ===> F14D03                             POSITION ===> 8
BYTES    ===> 8         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C14D03
-------------   10 ------------------------------------------------------------
SEGNAME  ===> S15
FLDNAME  ===> F15D01                             POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C15D01
-------------   11 ------------------------------------------------------------
SEGNAME  ===> S15
FLDNAME  ===> F15D02                             POSITION ===> 3
BYTES    ===> 30        DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C15D02
```

Figure 491. EKYMP53E: Entity and Extension Segments Fields/Columns (Example 2 - PR3) (Part 3 of 3)

Type END on the command line and press Enter (or press F3) to return to panel EKYMP52E (shown in Figure 487 on page 634). Repeat the procedure to return to panel EKYMP50E (shown in Figure 485 on page 633) with the state of options 1 and 2 changed to `processed`.

Type END on the command line and press Enter (or press F3) to return to panel EKYMP15E (shown in Figure 482 on page 632) with the state of options 1, 2, and 3 changed to `processed`.

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. Panel EKYMP15E is overlaid with the panel shown in Figure 492 on page 639.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

     _____DPROP MVGIN - Generate MVG Input Tables Insert Job_______  (name)
    | EKYMPIIE ==>                                                  |  3
    |                                                               |
    |                                 DPROP-System:        (name)   |  s
 1  |                                 Propagation Request: PR3      |  d *
 2  |                                                               |  d *
 3  |  Enter the member name to be used to store the generated      |  d *
    |  MVG input tables insert job.  The job is stored in data set  |  d *
 4  |  "(library)".                                                 |  d *
    |                                                               |
 5  |     Member Name    ===> PR3                                   |
    |                                                               |
 S  | Press <ENTER> to continue or enter END command to return.     |
    |                                                               |
 E  |                                                               |
    |_______________________________________________________________|


Enter option or move cursor before desired option.
 Enter END command to return.
```

Figure 492. Generate MVG Input Tables Insert Job (Example 2 - PR3)

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Providing MVGU control statements: 2

Figure 493 describes the IMS DPROP MVGU control statement that tells the MVGU to create PR3 in the IMS DPROP directory based on data stored in the MVG input tables.

```
CREATE PR=PR3
```

Figure 493. MVGU control statements to Create the Propagation Request (Example 2 - PR3)

Defining Propagation Requests in the MVG Input Tables with SQL Insert Statements: 2

Figure 494 on page 641 describes SPUFI statements that define propagation requests in the MVG input tables.

Typically, propagation requests are described in the MVG input tables by a program you have provided that has a dictionary as input. It is unusual to describe propagation requests in the MVG input tables with SPUFI statements.

As shown in Figure 494 on page 641, the following rows are inserted into the MVG input tables:

- One DPRIPR row containing general information about PR3
- One DPRISEG row
  - For each physical ancestor/parent segment of the entity segment (one row for the physical ancestor S11 and another row for the physical parent S12)
  - For the entity segment, S13
  - For each extension segment (one row for the extension segment S14 and another row for the extension segment S15).
- one DPRITAB row identifying the target table, T13
- one DPRIFLD row for each propagated field

The two key fields (F11KEY and F13KEY) of segments S11 and S13 are not propagated to two columns of the DB2 primary key. Instead, because of specific requirements of the application, they are propagated as four key subfields (F11K01, F11K02, F13K02, F13K04) to four columns of the DB2 primary key. Therefore, one row is inserted into the FLD table for each one of these four key subfields.

```
                                                                          NOTE

INSERT INTO T096606.DPRIPR
                   (PRID, PRTYPE, MAPCASE, MAPDIR,  ERROPT  , ACTION)
       VALUES      ('PR3', 'E'  , '2'    ,  'HR' , 'BACKOUT', 'REPL');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR3','D01','S11', 'P',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR3','D01','S12', 'P',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR3','D01','S13', 'E',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR3','D01','S14', 'X',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR3','D01','S15', 'X',' ');

INSERT INTO T096606.DPRITAB
                   (PRID ,TABQUAL, TABNAME)
       VALUES      ('PR3',' ','T13');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S11','F11K01',01,'C ',                  1
                    02,0,'      ',' ','T13','C11K01');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S11','F11K02',03,'P ',                  1,2
                    04,0,'      ',' ','T13','C11K02');


INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S12','F12K02',02,'P ',                  1,2
                    04,0,'      ',' ','T13','C12K02');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S13','F13K02',03,'P ',                  1,2
                    03,0,'      ',' ','T13','C13K02');
```

Figure 494. SPUFI Statements to Define PR3 in the MVG INput Tables (Example 2 - PR3) (Part 1 of 3)

```
INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S13','F13K03',06,'P ',                  1,2
                    05,0,'      ',' ','T13','C13K03');
INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S13','F13D01',01,'H ',
                    02,0,'      ',' ','T13','C13D01');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S13','F13D04',11,'H ',
                    02,0,'      ',' ','T13','C13D04');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S13','F13D05',13,'F ',
                    04,0,'      ',' ','T13','C13D05');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S13','F13D06',17,'H ',
                    02,0,'      ',' ','T13','C13D06');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S14','F14D01',01,'C ',
                    02,0,'      ',' ','T13','C14D01');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S14','F14D02',03,'C ',
                    05,0,'      ',' ','T13','C14D02');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S14','F14D03',08,'C ',
                    08,0,'      ',' ','T13','C14D03');
```

Figure 494. SPUFI Statements to Define PR3 in the MVG INput Tables (Example 2 - PR3) (Part 2 of 3)

```
INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S15','F15D01',01,'P ',
                    02,0,'      ',' ','T13','C15D01');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S15','F15D02',03,'C ',
                    30,0,'      ',' ','T13','C15D02');
```

Figure 494. SPUFI Statements to Define PR3 in the MVG INput Tables (Example 2 - PR3) (Part 3 of 3)

Notes:

1. The first five INSERT INTO T096606.DPRIFLD statements describe the mapping for the key fields and key subfields. They map the fully concatenated key of the entity segment to the columns of the DB2 primary key.

   The first two of these statements insert S11 into the SEGNAME column (because F11K01 and F11K02 are fields of S11, not S13).

   The third of these statements insert S12 into the SEGNAME column (because F12K02 is a field of S12, not S13).
2. The INSERT INTO T096606.DPRIFLD statement describes the mapping of IMS packed fields, F11K02, F12K02, F13K03, and F13K03 to the DB2 target columns. The value inserted into the BYTES column describes the length of the field in bytes (while the length of the column must be defined in the DB2 CREATE TABLE statement in digits).

Providing MVGU control statements: 2

Figure 495 describes the IMS DPROP MVGU control statement that tells the MVGU to create PR3 in the IMS DPROP directory based on data stored in the MVG input tables.

```
CREATE PR=PR3
```

Figure 495. MVGU control statements to Create the Propagation Request (Example 2 - PR3)

Example 3: Propagation with Mapping Case 3

In this example, an IMS segment contains two repeating group of fields that are propagated to two different tables by two propagation requests belonging to mapping case 3. In addition, fields of the IMS segment that do not belong to any repeating group of fields are propagated to a third table with a mapping case 1 propagation request.

The propagated IMS segment type, shown in Figure 496 on page 644, is called ACCOUNT and describes a bank account. This segment consists of:

- A first repeating group of fields describing different credit types associated with the bank account
- A second repeating group of fields describing different credit cards associated with the bank account

- Fields that do not belong to the repeating groups

In this example:

- Both repeating group of fields have a fixed length and a variable number of occurrences.
- The mapping case 3 propagation requests do not require a Segment exit routine.
- The IMS database is a root-only database.

Remember that propagation with mapping case 3 propagation requests requires a Segment exit routine if either:

- The propagation requests are TYPE=E Propagation Requests.
- The repeating group of fields do not contain ID fields uniquely identifying each occurrence of the repeating group, or
- The number of occurrences of an internal group is variable and this number is not stored in a field of the IMS segment.

An example of mapping case 3 propagation requests defined with a Segment exit routine is in IMS DPROP Customization Guide.

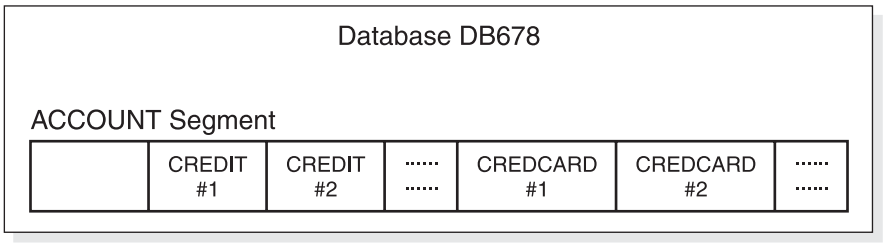

Figure 496. ACCOUNT Segment of IMS Database DB678 (Example 3)

Overview of the Performed Mapping and Propagation: 3

As recommended in the appropriate Administrators Guide for your propagation mode, the mapping is performed in the following two-step process:

1. Transformation into a normalized hierarchical structure
2. Mapping the normalized hierarchical structure to DB2

First, the IMS database hierarchy is conceptually transformed into a normalized hierarchical structure with a containing segment and internal segments. This is shown in Figure 497 on page 645.

Then, the mapping to DB2 is based on the conceptually normalized structure. This is shown in Figure 498 on page 646.

Transformation into a Normalized Hierarchical Structure

Figure 497 on page 645 shows how the IMS database is conceptually transformed into a normalized hierarchical structure.

- In the normalized hierarchical structure, the containing segment ACCOUNT consists of those fields that do not belong to any repeating group of fields.
- The internal segment CREDIT is the first repeating group of fields; the internal segment CREDCARD is the second repeating group of fields. The internal segments are children of the containing segment.

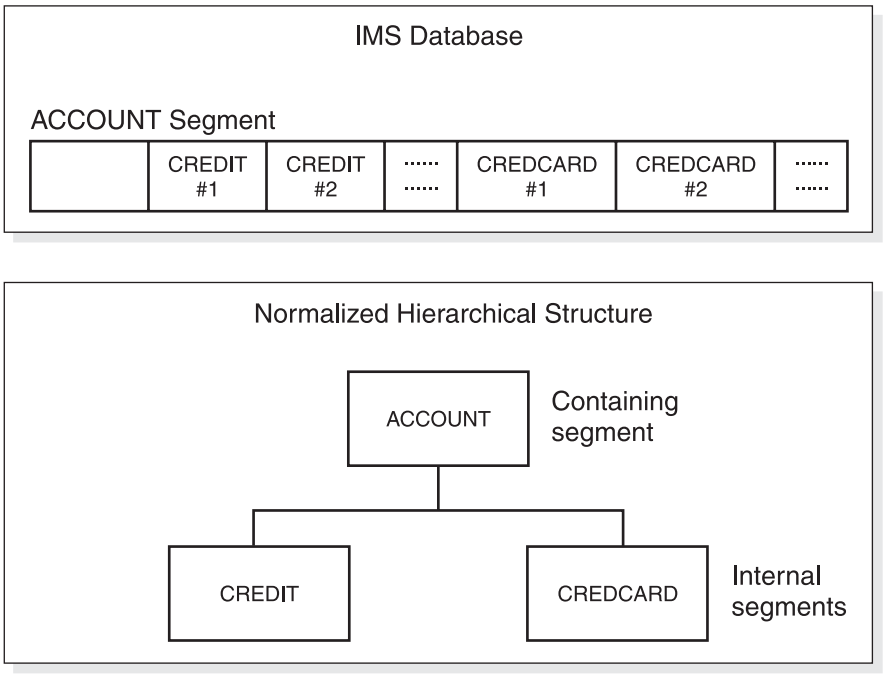

Figure 497. Conceptual Transformation into a Normalized Hierarchical Structure (Example 3)

Mapping the Normalized Hierarchical Structure to DB2

Figure 498 on page 646 illustrates the propagation of:

- The two internal segment types CREDIT and CREDCARD to the tables CREDIT and CREDCARD with two propagation requests belonging to mapping case 3.
- The containing segment type ACCOUNT to the table ACCOUNT with a propagation request belonging to mapping case 1.

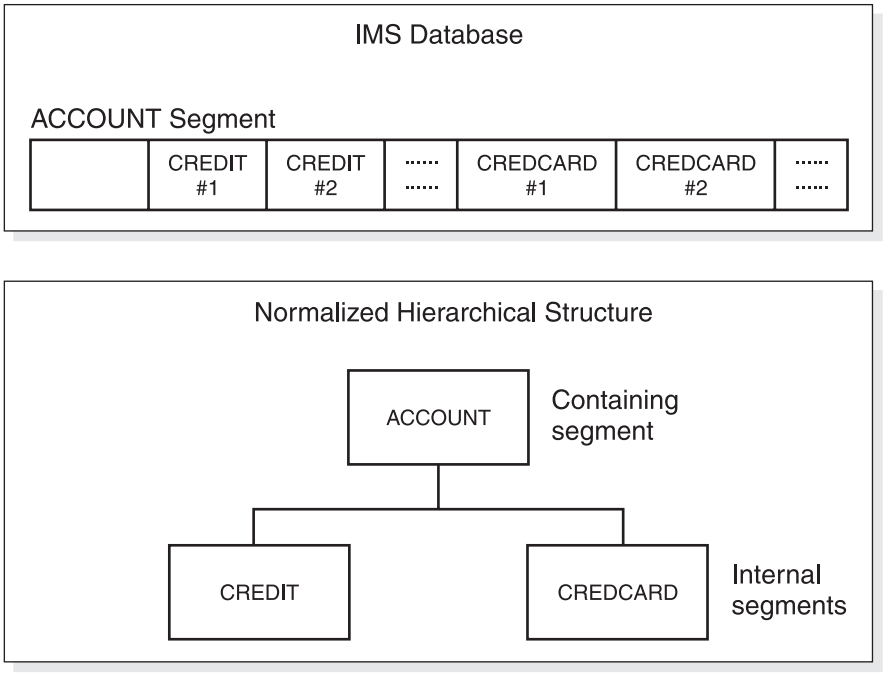

Figure 498. Overview of Propagation of Database DB678 (Example 3)

Figure 499 illustrates, for each propagation request, the mapping between the keys (and ID fields) of the segments and the DB2 primary keys of the tables.

This example assumes that both repeating group of fields have ID fields uniquely identifying each occurrence of the field group. If no such ID field exists, you must provide a Segment exit routine that constructs an artificial ID field within the edited format of each occurrence of the repeating group of fields.

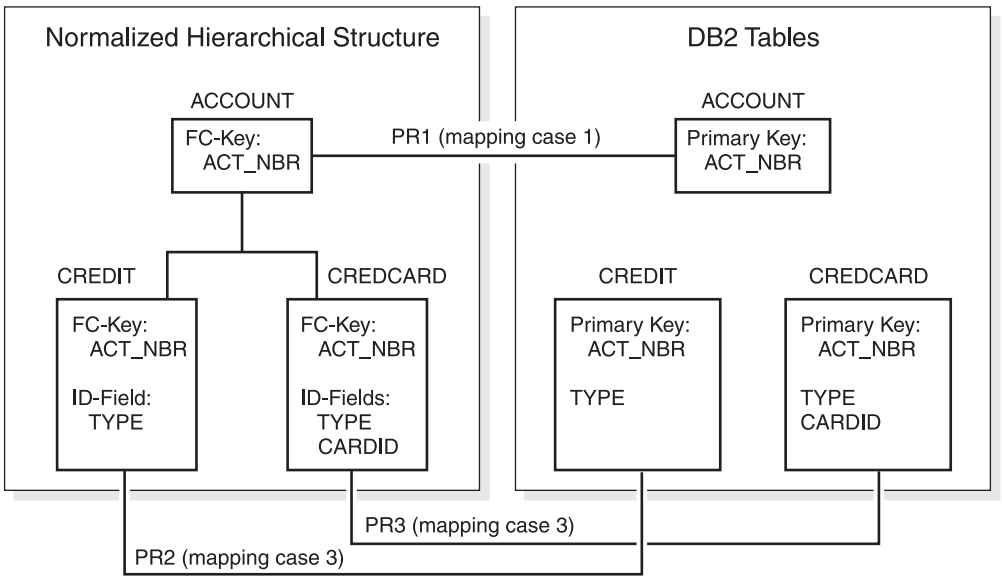

Figure 499. Key Mapping Between the Entity Segments and the Target Tables (Example 3)

Notes:

1. For the propagation request propagating the containing segment ACCOUNT, the fully concatenated key of the containing segment is mapped, as IMS DPROP recommends, to the DB2 primary key.

   The fully concatenated key of the segment ACCOUNT consists of the field ACT_NBR.
2. For the propagation requests mapping the internal segments CREDIT and CREDCARD, the key mapping is between:
   - The combination of the fully concatenated key (ACT_NBR) and ID fields identifying each occurrence of the internal segment (the ID field of the internal segment CREDIT is the field TYPE; the ID fields of the internal segment CREDCARD are the fields TYPE and CARDID), and
   - The columns of the DB2 primary keys (ACT_NBR, TYPE and CARDID).

Field Mapping Overview: 3

Table 49 shows for each propagation request the characteristics of the source IMS fields and the target DB2 columns. The table contains the following information:

- For the IMS source field:
  - The name of the IMS field (as defined to IMS DPROP either on the NAME keyword of the DataRefresher FIELD statement, or in the FLDNAME column of the DPRIFLD MVG input table)
  - The data type (as defined to IMS DPROP either on the TYPE keyword of the DataRefresher FIELD statement, or in the DATATYPE column of the DPRIFLD MVG input table)
  - The length in bytes (as defined to IMS DPROP either on the BYTES keyword of the DataRefresher FIELD statement or in the BYTES column of the DPRIFLD MVG input table)
- For the DB2 target column:
  - The name of the target DB2 column (as defined to DB2 in the CREATE TABLE statement)
  - The data type of the target DB2 column (as defined to DB2 in the CREATE TABLE statement)

Table 49. Source Fields and Target Columns

Source IMS fields				**Target DB2 columns**	
Field Name	**See Note Below**	**Data Type**	**Length (in Bytes)**	**Column Name**	**Data Type**
			PR1		
ACT_NBR	1	Z	9	ACT_NBR	DECIMAL(9,0)
NAME		C	21	NAME	CHAR(21)
COUNT1	3	H	2	-	
COUNT2	3	H	2	-	
			PR2		
ACT_NBR	1	Z	9	ACT_NBR	DECIMAL(9,0)
COUNT1	3	H	2	-	
TYPE	2	P	1	TYPE	DECIMAL(1,0)
AMOUNT		P	7	AMOUNT	DECIMAL(15,2)
LIMIT		P	7	LIMIT	DECIMAL(15,2)

Table 49. Source Fields and Target Columns (continued)

Source IMS fields				Target DB2 columns	
Field Name	See Note Below	Data Type	Length (in Bytes)	Column Name	Data Type
			PR3		
ACT_NBR	1	Z	9	ACT_NBR	DECIMAL(9,0)
COUNT2	3	H	2	-	
TYPE	2	C	10	TYPE	CHAR(10)
CARDID	2	C	10	CARDID	CHAR(10)
EXPIRES		A	10	EXPIRES	DATE

Notes:

1. ACT_NBR is the fully concatenated key of the containing IMS segment ACCOUNT. It is propagated to a column of the DB2 primary key of the target tables.
2. TYPE and CARDID are ID fields located in the internal segments. They uniquely identify each occurrence of the internal segments, and are propagated to columns of the DB2 primary key of the target tables of the mapping case 3 Propagation Requests.
3. COUNT1 and COUNT2 are the count fields containing the number of occurrences of the internal segments CREDIT and CREDCARD. As in most cases, they are not propagated to a target table.

   In other examples with a variable number of internal segment occurrences, the containing IMS segment could have no count fields. In this case, you must provide a Segment exit routine that constructs the count fields in the edited format of the segment.

DBD Definitions: 3

The DBD control statements for the model database are shown in Figure 500.

```
         DBD NAME=DB678,VERSION=V129,                                C
               ACCESS=(HDAM,OSAM),RMNAME=(DFSHDC40,5,4),             C
               EXIT=(EKYRUP00)
         DATASET DD1=HDAM,SIZE=4096,DEVICE=3380
*
         SEGM  NAME=ACCOUNT,PARENT=0,BYTES=(381,150)
         FIELD NAME=(ACNTNBR,SEQ,U),BYTES=9,START=3
*
         DBDGEN
         FINISH
         END
```

Figure 500. Control statements for Model Database DB678 (Example 3)

The EXIT keyword in the DBD control statement tells IMS to give control to EKYRUP00, the IMS DPROP-provided DL/I Data Capture Exit routine, when a DL/I call updates a segment of the database DB678.

Usually, IMS DPROP and DataRefresher do not require you to define the non-key fields in the IMS DBD. In this example, the non-key fields are not defined in the IMS DBD.

Table Definitions: 3

Figure 501 describes the SQL statements to create the target tables ACCOUNT, CREDIT, and CREDCARD. As required by IMS DPROP, these tables are defined with a DB2 primary key.

```
CREATE TABLE ACCOUNT
          (ACT_NBR   DECIMAL (9,0)  NOT NULL,
           NAME      CHAR(21)       NOT NULL WITH DEFAULT,
       PRIMARY KEY
          (ACT_NBR))
       IN DU096606.PROPT1    ;
CREATE UNIQUE INDEX XN01 ON ACCOUNT
          (ACT_NBR)
       USING VCAT KOE ;


CREATE TABLE CREDIT
          (ACT_NBR   DECIMAL (9,0)   NOT NULL,
           TYPE      DECIMAL (1,0)   NOT NULL,
           AMOUNT    DECIMAL (15,2)  NOT NULL WITH DEFAULT,
           LIMIT     DECIMAL (15,2)  NOT NULL WITH DEFAULT,
       PRIMARY KEY
          (ACT_NBR, TYPE))
       IN DU096606.PROPT2    ;
CREATE UNIQUE INDEX XN02 ON CREDIT
          (ACT_NBR, TYPE)
       USING VCAT KOE ;


CREATE TABLE CREDCARD
          (ACT_NBR   DECIMAL (9,0)   NOT NULL,
           TYPE      CHAR(10)        NOT NULL,
           CARDID    CHAR(10)        NOT NULL,
           EXPIRES   DATE                    ,
       PRIMARY KEY
          (ACT_NBR, TYPE, CARDID))
       IN DU096606.PROPT3    ;
CREATE UNIQUE INDEX XN03 ON CREDCARD
          (ACT_NBR, TYPE, CARDID)
       USING VCAT KOE ;
```

Figure 501. Tables ACCOUNT, CREDIT, and CREDCARD (Example 3)

Defining Propagation Requests with DataRefresher: 3

Defining the DXTPSB: 3

DataRefresher requires that you describe the IMS database to DataRefresher by providing a CREATE DXTPSB command. The DataRefresher control statements for our model database are shown in Figure 502 on page 650.

The definition includes three SEGMENT commands:

- One describing the IMS segment ACCOUNT
- Two describing the internal segments CREDIT and CREDCARD.

Because the internal segments (repeating group of fields) have a variable number of occurrences, they must have a count field. The two count fields, *COUNT1* and *COUNT2*, are described on FIELD commands of the ACCOUNT segment.

Control statements that are highlighted numerically (**1** or **2**, for example) are further described below the figure.

```
                                                                    NOTE

CREATE DXTPSB    NAME=KOEPSB2

       DXTPCB    NAME=PCB001,DBACCESS=HDAM,DBNAME=DB678

         SEGMENT NAME=ACCOUNT , PARENT=0, BYTES=381, FORMAT=V          1

             FIELD    NAME=KEY    , START=3,  BYTES=9 , SEQFLD=R       2
             FIELD    NAME=ACT_NBR, START=3,  BYTES=9 , TYPE=Z         2
             FIELD    NAME=NAME   , START=12, BYTES=21, TYPE=C
             FIELD    NAME=COUNT1 , START=33, BYTES=2 , TYPE=H         3
             FIELD    NAME=COUNT2 , START=35, BYTES=2 , TYPE=H         4

         SEGMENT NAME=CREDIT  , PARENT=ACCOUNT ,
                 FORMAT=FI,                                           5
                 OCCURS=COUNT1,                                       3
                 START =37,                                           6
                 BYTES =15                                            7

             FIELD    NAME=TYPE  , START=1,  BYTES=1 , TYPE=P, SCALE=0  8
             FIELD    NAME=AMOUNT, START=2,  BYTES=7,  TYPE=P, SCALE=2
             FIELD    NAME=LIMIT , START=9,  BYTES=7,  TYPE=P, SCALE=2

         SEGMENT NAME=CREDCARD, PARENT=ACCOUNT ,
                 FORMAT=FI,                                           5
                 OCCURS=COUNT2,                                       4
                 START =CREDIT+1,                                     9
                 BYTES =30                                            10

             FIELD    NAME=TYPE    , START=1,  BYTES=10, TYPE=C        8
             FIELD    NAME=CARDID  , START=11, BYTES=10, TYPE=C        8
             FIELD    NAME=EXPIRES , START=21, BYTES=10, TYPE=A ;
```

Figure 502. DataRefresher control statements

Notes:

1. The SEGMENT statement with the ACCOUNT name defines the IMS segment to DataRefresher. FORMAT=V tells DataRefresher that the IMS segment has a variable length.

   The segment name defined to DataRefresher must match the segment name defined to IMS in the DBD.
2. KEY is the IMS key field (as defined in the IMS DBDGEN) of the root segment S11. This field is defined to DataRefresher with a SEQFLD=R keyword as the IMS key field.

   This key field is redefined to DataRefresher as a zoned numeric field called ACT_NBR.
3. COUNT1 is the numerical field containing the actual number of occurrences of the internal segment CREDIT.

   Remember that in this example, the internal segments have a variable number of occurrences and must have an occurrence field.

   If the number of occurrences is fixed, no occurrence field is needed; in this case, the fixed number of occurrences is specified on the OCCURS= operand of the SEGMENT statement describing the internal segment.

4. COUNT2 is the numerical field containing the actual number of occurrences of the internal segment CREDCARD.
5. The SEGMENT statements with the CREDIT and CREDCARD names define the two internal segments (the two repeating group of fields) to DataRefresher.

   FORMAT=FI tells DataRefresher that the segment is a fixed-length internal segment.
6. START=37 tells DataRefresher that the first occurrence of the internal segment CREDIT has a fixed start-position within the IMS segment, the 37th byte.

   The first occurrence of an internal segment type could have a variable start-position. In this case, the START= operand is coded either as START=fieldname+n or as START=segname+n.
7. BYTES=15 tells DataRefresher that the fixed length of the segment is 15 bytes.

   The internal segment could have a variable length. In this case, the length of an internal segment is indirectly specified through the NEXT=fieldname+n operand.
8. The fields TYPE and CARDID are ID fields uniquely identifying each occurrence of an internal segment within the containing IMS segment.

   If the repeating group of fields/internal segments do not have ID fields, you must provide a Segment exit routine that constructs the ID fields in the edited format of the segment.
9. START=CREDIT+1 tells DataRefresher that the first occurrence of the internal segment CREDCARD has a variable position. START=CREDIT+1 specifies that the start position is located immediately after the end of the last occurrence of the CREDIT internal segment/repeating group of fields.
10. BYTES=30 tells DataRefresher that the internal segment CREDCARD has a fixed length of 30 bytes.

Defining the DXTVIEW: 3

DataRefresher requires a view description for each internal segment. Because our example has two internal segments, two DXTVIEWs must be defined to DataRefresher.

The DataRefresher commands used to define the views for our example are shown in Figure 503.

```
                                                    NOTE
CREATE
   DXTVIEW NAME     = VIEWCRED,
           DXTPSB   = KOEPSB2,                      1
           DXTPCB   = PCB001,                       2
           SEGMENT  = CREDIT,                       3
           MINSEGM  = CREDIT,                       3
           FIELDS   = *         ;                   4

CREATE
   DXTVIEW NAME     = VIEWCARD,
           DXTPSB   = KOEPSB2,                      1
           DXTPCB   = PCB001,                       2
           SEGMENT  = CREDCARD,                     3
           MINSEGM  = CREDCARD,                     3
           FIELDS   = *         ;                   4
```

Figure 503. VIEWCRED and VIEWCARD for the Two Internal Segments (Example 3)

Notes:

1. The DXTPSB=KOEPSB2 keyword identifies the previously created DXTPSB.
2. The DXTPCB=PCB001 keyword identifies the DXTPCB of the DXTPSB that describes the propagated IMS database.
3. For DXTVIEWs used with PRs belonging to mapping case 3, SEGMENT= and MINSEGM= identify the internal segment.
4. The FIELDS=* operand tells DataRefresher that all fields of the described hierarchical path should be included in the DXTVIEW.

Providing DataRefresher UIM SUBMIT Commands and EXTRACT Statements: 3

Figure 504 on page 653 illustrates the DataRefresher SUBMIT commands and EXTRACT statements used to define the propagation requests that will propagate the containing segment ACCOUNT and the internal segments CREDIT and CREDCARD.

The DataRefresher SUBMIT/EXTRACT creates the propagation requests in the IMS DPROP directory but does not load the data into the DB2 tables. This is the responsibility of the DataRefresher DEM. Refer to “Providing DataRefresher DEM control statements: 3 ” on page 654 for examples of a DataRefresher DEM control statement.

```
                                                                    NOTE
SUBMIT   EXTID=PR1,
               NODE=NODEX,
               USERID=T096606,
               CD=JCS,                                              2
               JCS=DDJCS01,                                         3
               FORMAT=SOURCE,                                       1
               USERDECK='ENFORCE NO, REPLACE',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=L,MAPDIR=HR,MAPCASE=1,
                         ACTION=REPL, ERROPT=BACKOUT'
EXTRACT INTO ACCOUNT (ACT_NBR   NOT NULL,                           4
                      NAME      NOT NULL WITH DEFAULT)
        OPTIONS(FLDERR(SUBST(ZERO)))
        SELECT        ACT_NBR,
                      NAME
        FROM VIEWCRED ;                                             5


SUBMIT   EXTID=PR2,
               NODE=NODEX,
               USERID=T096606,
               CD=JCS,                                              2
               JCS=DDJCS01,                                         3
               FORMAT=SOURCE,                                       1
               USERDECK='ENFORCE NO, REPLACE',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=L,MAPDIR=HR,MAPCASE=3,
                         ACTION=REPL, ERROPT=BACKOUT'
EXTRACT INTO CREDIT (ACT_NBR   NOT NULL,                            4
                     TYPE      NOT NULL,
                     AMOUNT    NOT NULL WITH DEFAULT,
                     LIMIT     NOT NULL WITH DEFAULT)
        OPTIONS(FLDERR(SUBST(ZERO)))
        SELECT       ACT_NBR,
                     TYPE   ,
                     AMOUNT ,
                     LIMIT
        FROM VIEWCRED ;                                             5

SUBMIT   EXTID=PR3,
               NODE=NODEX,
               USERID=T096606,
               CD=JCS,                                              2
               JCS=DDJCS01,                                         3
               FORMAT=SOURCE,                                       1
               USERDECK='ENFORCE NO, REPLACE',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=L,MAPDIR=HR,MAPCASE=3,
                         ACTION=REPL, ERROPT=BACKOUT'
EXTRACT INTO CREDCARD (ACT_NBR   NOT NULL,                          4
                       TYPE      NOT NULL,
                       CARDID    NOT NULL,
                       EXPIRES   )
        OPTIONS(FLDERR(SUBST(ZERO)))
        SELECT       ACT_NBR,
                     TYPE   ,
                     CARDID,
                     EXPIRES
        FROM VIEWCARD ;                                             6
```

Figure 504. DataRefresher SUBMIT/EXTRACT for PR1, PR2, and PR3 (Example 3)

Notes:

1. As recommended by IMS DPROP, FORMAT=SOURCE is specified to reduce overhead during the extract.
2. CD=JCS is specified, which tells DataRefresher to create a control deck for the DB2 Load utility. DataRefresher DEM creates the control deck as a SYSIN file.
3. JCS=DDJCS01 is specified, which tells DataRefresher that you are providing JCL statements for the DB2 Load utility in the file allocated through a //DDJCS01 DD statement.
4. The INTO clause of the EXTRACT statement provides the same NOT NULL/NOT NULL WITH DEFAULT specifications as the DB2 CREATE TABLE statement. This ensures that the mapping performed by DataRefresher during the extract is compatible to the mapping performed by IMS DPROP during propagation.
5. VIEWCRED is the previously created DXTVIEW for one of the two internal segments.
6. VIEWCARD is the previously created DXTVIEW for the second internal segment.

Providing DataRefresher DEM control statements: 3

Figure 505 shows an example of a DataRefresher DEM control statement that you can use to perform the DataRefresher extract for all propagation requests using KOEPSB2 as DXTPSB.

```
INITDEM   NAME=JUDEM03
USE DXTPSB=KOEPSB2;
```

Figure 505. Sample DXT DEM control statement

Defining Propagation Requests with the MVGIN Front End Application: 3

In this example, three different propagation requests are defined on the MVG input tables. The entire panel flow is shown for each propagation request.

Call the MVGIN Front End Application as described in Chapter 20, “CCU Front End ,” on page 523. The panel shown in Figure 506 on page 655 is displayed.

```
EKYMP00E ------------- DPROP MVG Input Tables Application ---------------------
Command ===>

                                                              DPROP Environ-
   1 - Insert, Update, Show, Delete a Propagation Request     ment  : (name)
   2 - Propagation Request Overview to Update, Show,          DPROP-
       Delete a Propagation Request                           System: (name)
   3 - Setup IMS and DPROP System Environment                 DPROP-System-
   4 - Edit existing MVG Input Tables Insert/Delete Jobs      Number: (number)

   E - Return


     Enter the fully qualified name of the partitioned data set to be
     used to store the generated MVGIN job control:
        Data Set Name ===>
     Enter the volume serial name to be used to allocate a new data set:
        Volume Serial ===>


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 506. EKYMP00E: DPROP MVG Input Tables Application (Example 3)

The first time you call the MVGIN Front End Application, you have to complete the setup panels and enter a data set name in the Data Set Name field.

Panel Flow for PR1: 3

The panel shown in Figure 507 is displayed.

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                       DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR1


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Figure 507. EKYMP10E: Maintain a Propagation Request (Example 3)

Type I in the Option field and PR1 in the propagation request Name field and press Enter. After your input is validated, the panel shown in Figure 508 on page 656 is displayed.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR1

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 508. EKYMP15E: Propagation Request Definition Selection (Example 3 - PR1)

Select option 1 to enter the general propagation request specifications. The panel shown in Figure 509 is displayed.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------
Command ===>
                                                Propagation Request: PR1

                                                                    More:   - +
  PR Type                   ===> L         ( E, L, F or U                   )
  Mapping Case              ===> 1         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===>           ( ID, DENORM or blank            )
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <
  User ID                   ===> T096606   ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 509. EKYMP20E: Propagation Request (Example 3 - PR1)

Type END and press Enter to return to panel EKYMP15E (shown in Figure 508 on page 656) with the state of option 1 changed to `processed`.

Select option 2 on panel EKYMP15E to define the DB2 table. The panel shown in Figure 510 is displayed.

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                   Propagation Request: PR1

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===> T096606     ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> ACCOUNT              ( a name up to 18 bytes )



         Note: The DB2 table you enter is not checked for existence.







Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 510. EKYMP30E: DB2 Table (Example 3 - PR1)

Type END and press Enter to return to panel EKYMP15E (shown in Figure 508 on page 656) with the state of options 1 and 2 changed to `processed`.

Select option 3 on panel EKYMP15E to define the segments, fields and columns. The panel shown in Figure 511 is displayed.

```
EKYMP40E --- DPROP MVGIN - Entity Segment for a Mapping Case 1 PR -------------
Command ===>

    F - Fields for the entity segment            Propagation Request: PR1
    P - Fields for the parent segments           Database-
                                                 Organization: HDAM/OSAM

  IMS Database Name         ===> DB678     ( a name up to 8 bytes           )
  Segment Name              ===> ACCOUNT   ( a name up to 8 bytes           )


  PCB Label Name            ===>           ( a name up to 8 bytes           )


  Segment Exit Routine      ===>           ( a name up to 8 bytes, optional )
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760             )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                         )


Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 511. EKYMP40E: Entity Segment for a Mapping Case 1 Propagation Request (Example 3 - PR1)

Because the ACCOUNT segment is the root segment, you cannot define fields for parent segments.

To define the fields for the entity segment, type F on the command line of panel EKYMP40E and press ENTER. The panel shown in Figure 512 is displayed.

```
EKYMP41E -------- DPROP MVGIN - Entity Segment Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR1
    S - Sort in POSITION sequence                Entity Segment:      ACCOUNT

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
FLDNAME  ===> ACT_NBR                            POSITION ===> 3
BYTES    ===> 9         DATATYPE ===> Z          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> ACT_NBR
-------------    2 ------------------------------------------------------------
FLDNAME  ===> NAME                               POSITION ===> 12
BYTES    ===> 21        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> NAME
-------------    3 ------------------------------------------------------------
FLDNAME  ===> COUNT1                             POSITION ===> 33
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    4 ------------------------------------------------------------
FLDNAME  ===> COUNT2                             POSITION ===> 35
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
```

Figure 512. EKYMP41E: Entity Segment Fields/Columns (Example 3 - PR1)

The count fields COUNT1 and COUNT2 contain the number of occurrences of the repeating group of fields within the IMS segment. Because count fields are not propagated, the values of the COLNAME fields are blank.

Type END and press Enter to return to panel EKYMP40E (shown in Figure 511 on page 657). Repeat the procedure to return to panel EKYMP15E (shown in Figure 508 on page 656) with the state of options 1, 2, and 3 changed to processed.

Select option 5 on panel EKYMP15E (shown in Figure 508 on page 656) to generate the MVG input tables insert job. The panel shown in Figure 513 on page 659 is overlaid on panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

        _____DPROP MVGIN - Generate MVG Input Tables Insert Job_______    ame)
       | EKYMPIIE ==>                                                 |  1
       |                                                              |
       |                                DPROP-System:        (name)   |  s
    1  |                                Propagation Request: PR1      |  d *
    2  |                                                              |  d *
    3  |  Enter the member name to be used to store the generated     |  d *
       |  MVG input tables insert job.  The job is stored in data set |  d *
    4  |  "(library)".                                                |  d *
       |                                                              |
    5  |     Member Name    ===> PR1                                  |
       |                                                              |
    S  | Press <ENTER> to continue or enter END command to return.    |
       |                                                              |
    E  |                                                              |
       |_____________________________________________________________|


Enter option or move cursor before desired option.
 Enter END command to return.
```

Figure 513. EKYMP15E: Propagation Request Definition Selection (Example 3 - PR1)

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Return to panel EKYMP10E to define propagation request PR2.

Panel Flow for PR2: 3

After creating PR1, the panel shown in Figure 514 is displayed.

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                        DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR2


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                 ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Figure 514. EKYMP10E: Maintain a Propagation Request (Example 3 - PR2)

Type I in the Option field and PR2 in the propagation request Name field and press Enter. After your input is validated, the panel shown in Figure 515 on page 660 is displayed.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR2

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 515. EKYMP15E: Propagation Request Definition Selection (Example 3 - PR2)

Select option 1 to enter the general propagation request specifications. The panel shown in Figure 516 is displayed.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------
Command ===>
                                                Propagation Request: PR2

                                                                    More:   - +
  PR Type                   ===> L         ( E, L, F or U                   )
  Mapping Case              ===> 3         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===>           ( ID, DENORM or blank            )
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <
  User ID                   ===> T096606   ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 516. EKYMP20E: Propagation Request (Example 3 - PR2)

Type END and press Enter to return to panel EKYMP15E (shown in Figure 515 on page 660) with the state of option 1 changed to `processed`.

Select option 2 on panel EKYMP15E to define the DB2 table. The panel shown in Figure 517 is displayed.

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                  Propagation Request: PR2

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===> T096606     ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> CREDIT               ( a name up to 18 bytes )



         Note: The DB2 table you enter is not checked for existence.







Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 517. EKYMP30E: DB2 Table (Example 3 - PR2)

Type END and press Enter to return to panel EKYMP15E (shown in Figure 515 on page 660) with the state of options 1 and 2 changed to `processed`.

Select option 3 on panel EKYMP15E to define the segments, fields and columns. The panel shown in Figure 518 is displayed.

```
EKYMP60E - DPROP MVGIN - Segments and Fields for a Mapping Case 3 PR ----------
Command ===>


                                                  Propagation Request: PR2

                                        State
    1 - Containing Segment/         not processed
        Internal Segment and
        Fields/Columns              not processed

    2 - Start Segments              not processed
        ( optional )


    E - Return




Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 518. EKYMP60E: Segments and Fields for a Mapping Case 3 Propagation Request (Example 3 - PR2)

Select option 1 to define the containing segment, the internal segment and the fields/columns. The panel shown in Figure 519 is displayed.

```
EKYMP61E ------- DPROP MVGIN - Containing and Internal Segment ----------------    NOTE
Command ===>

    F - Fields for the internal segment          Propagation Request: PR2
    P - Fields for the parent segments and       Database-
        for the containing segment               Organization: HDAM/OSAM

                                                                More:   - +
     ***  Containing Segment  ***
  IMS Database Name         ===> DB678     ( a name up to 8 bytes          )
  Segment Name              ===> ACCOUNT   ( a name up to 8 bytes          )

  PCB Label Name            ===>           ( a name up to 8 bytes          )

  Segment Exit Routine      ===>
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760            )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                        )

     ***   Internal Segment   ***
  Segment Name              ===> CREDIT    ( a name up to 8 bytes          )
  Segment Format            ===> FI        ( FI or VI                      )      1
  Number of Occurrences of the Segment ( number or fieldname )
                            ===> COUNT1                                           2
  Start                     ===> 37                                             3
  Segment Length            ===> 15        ( from 1 up to 32760            )      4
  Next Occurrence ( fieldname+n )
                            ===>

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 519. EKYMP61E: Containing and Internal Segment (Example 3 - PR2)

Notes:

1. The Segment Format field is FI, identifying the segment as a fixed-length internal segment.
2. The Number of Occurrences of the Segment field is COUNT1. The internal segment has a variable number of occurrences, and the number of occurrences is stored in the field COUNT1.

   If the number of occurrences is fixed, an occurrence field is not needed; in this case, the fixed number of occurrences must be provided as a value in the Number of Occurrences of the Segment field.
3. The Start field is 37, indicating that the first occurrence of the internal segment has a fixed start position within the containing IMS segment, the 37th byte.

   The first occurrence of an internal segment type could have a variable start-position. In this case, the value of the Start field is coded either as *fieldname+n* or as *segname+n*.
4. The Segment Length field is 15, indicating that the fixed length of the internal segment is 15 bytes.

   The internal segment can have a variable length. In this case, the length of an internal segment is indirectly specified by *fieldname+n* in the Next Occurrence field.

To define the fields for the parent and the containing segments, type P on the command line and press ENTER. The panel shown in Figure 520 on page 663 is displayed.

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------    NOTE
Command ===>                                                   SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR2
        sequence                             Entity/Internal Segment: CREDIT

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> ACCOUNT     Segment Level:
FLDNAME  ===> ACT_NBR                            POSITION ===> 3
BYTES    ===> 9         DATATYPE ===> Z          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> ACT_NBR
-------------    2 ------------------------------------------------------------
SEGNAME  ===> ACCOUNT     Segment Level:
FLDNAME  ===> COUNT1                             POSITION ===> 33
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>                                                                    1
```

Figure 520. EKYMPPFE: Parent Segments Fields/Columns (Example 3 - PR2)

Note: The COLNAME field is blank because the count field COUNT1 is not propagated.

Type END on the command line and press Enter to return to panel EKYMP61E (shown in Figure 519 on page 662).

To define the fields for the entity segment, enter F on the command line of panel EKYMP61E and press Enter. The panel shown in Figure 521 on page 664 is displayed.

```
EKYMP62E ------- DPROP MVGIN - Internal Segment Fields/Columns ----------------
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                    Propagation Request: (name)
    S - Sort in POSITION sequence                Internal Segment:    CREDIT

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
FLDNAME  ===> TYPE                               POSITION ===> 1
BYTES    ===> 1         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> TYPE
-------------    2 ------------------------------------------------------------
FLDNAME  ===> AMOUNT                             POSITION ===> 2
BYTES    ===> 7         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> AMOUNT
-------------    3 ------------------------------------------------------------
FLDNAME  ===> LIMIT                              POSITION ===> 9
BYTES    ===> 7         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> LIMIT
```

Figure 521. EKYMP62E: Internal Segment Fields/Columns (Example 3 - PR2)

Type END on the command line and press Enter to return to panel EKYMP61E (shown in Figure 519 on page 662). Repeat the procedure to return to panel EKYMP60E. (shown in Figure 518 on page 661).

Because this propagation request does not have any start segments, you do not have to process option 2.

Type END on the command line and press Enter to return to panel EKYMP15E (shown in Figure 515 on page 660) with the state of options 1, 2, and 3 changed to `processed`.

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. The panel shown in Figure 522 on page 665 is overlaid on panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

      _______PROP MVGIN - Generate MVG Input Tables Insert Job_______   (name)
     | EKYMPIIE ==>                                                  |   2
     |                                                               |
     |                                DPROP-System:        (name)    |   s
  1  |                                Propagation Request: PR2       |   d *
  2  |                                                               |   d *
  3  |  Enter the member name to be used to store the generated      |   d *
     |  MVG input tables insert job.  The job is stored in data set  |   d *
  4  |  "(library)".                                                 |   d *
     |                                                               |
  5  |     Member Name    ===> PR2                                   |
     |                                                               |
  S  | Press <ENTER> to continue or enter END command to return.     |
     |                                                               |
  E  |                                                               |
     |_______________________________________________________________|


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 522. Generate MVG Input Tables Insert Job

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Return to panel EKYMP10E to define propagation request PR3.

Panel Flow for PR3: 3

After creating PR1, the panel shown in Figure 523 is displayed.

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                      DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR3


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                 ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Figure 523. EKYMP10E: Maintain a Propagation Request (Example 3 - PR3)

Type I in the Option field and PR3 in the propagation request Name field and press Enter.

After your input is validated, the panel shown in Figure 524 on page 666 is displayed.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR3

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 524. EKYMP15E: Propagation Request Definition Selection (Example 3 - PR3)

Select option 1 to enter the general propagation request specifications. The panel shown in Figure 525 EKYMP20E is displayed.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------
Command ===>
                                                Propagation Request: PR3

                                                                      More:   - +
  PR Type                   ===> L         ( E, L, F or U                   )
  Mapping Case              ===> 3         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===>           ( ID, DENORM or blank            )
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <
  User ID                   ===> T096606   ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 525. EKYMP20E: Propagation Request (Example 3 - PR3)

Type END and press Enter to return to panel EKYMP15E (shown in Figure 524 on page 666) with the state of option 1 changed to `processed`.

Select option 2 on panel EKYMP15E to define the DB2 table. The panel shown in Figure 526 is displayed.

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                   Propagation Request: PR3

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===> T096606     ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> CREDCARD             ( a name up to 18 bytes )



         Note: The DB2 table you enter is not checked for existence.







Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 526. EKYMP30E: DB2 Table (Example 3 - PR3)

Type END on the command line and press Enter to return to panel EKYMP15E (shown in Figure 524 on page 666) with the state of options 1 and 2 changed to `processed`.

Select option 3 on panel EKYMP15E to define the segments, fields and columns. The panel shown in Figure 527 is displayed.

```
EKYMP60E - DPROP MVGIN - Segments and Fields for a Mapping Case 3 PR ----------
Command ===>


                                                   Propagation Request: PR3

                                        State
    1 - Containing Segment/         not processed
        Internal Segment and
        Fields/Columns              not processed

    2 - Start Segments              not processed
        ( optional )


    E - Return




Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 527. EKYMP60E: Segments and Fields for a Mapping Case 3 Propagation Request (Example 3 - PR3)

Select option 1 to define the containing segment, the internal segment and the fields and columns. The panel shown in Figure 528 is displayed.

```
EKYMP61E ------- DPROP MVGIN - Containing and Internal Segment ----------------    NOTE
Command ===>

    F - Fields for the internal segment          Propagation Request: PR3
    P - Fields for the parent segments and       Database-
        for the containing segment               Organization: HDAM/OSAM

                                                                More:   - +
     ***  Containing Segment  ***
  IMS Database Name         ===> DB678     ( a name up to 8 bytes          )
  Segment Name              ===> ACCOUNT   ( a name up to 8 bytes          )

  PCB Label Name            ===>           ( a name up to 8 bytes          )

  Segment Exit Routine      ===>
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760            )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                        )

     ***   Internal Segment   ***
  Segment Name              ===> CREDCARD  ( a name up to 8 bytes          )
  Segment Format            ===> FI        ( FI or VI                      )      1
  Number of Occurrences of the Segment ( number or fieldname )
                            ===> COUNT2                                           2
  Start                     ===> CREDIT+1                                         3
  Segment Length            ===> 30        ( from 1 up to 32760            )      4
  Next Occurrence ( fieldname+n )
                            ===>

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 528. EKYMP61E: Containing and Internal Segment (Example 3 - PR3)

Notes:

1. The Segment Format field is FI. The segment is a fixed-length internal segment.
2. The Number of Occurrences of the Segment field is COUNT2. The internal segment has a variable number of occurrences, and the number of occurrences is stored in the field COUNT2.

   If the number of occurrences is fixed, there is no need for an occurrence field; in this case, the fixed number of occurrences must be provided as a value in the Number of Occurrences of the Segment field.
3. The Start field is CREDIT+1. The first occurrence of the internal segment CREDCARD has a variable start position within the containing IMS segment. CREDIT+1 specifies that the start position is located immediately after the end of the last occurrence of the CREDIT internal segment/repeating group of fields.
4. The Segment Length field is 30. The fixed length of the internal segment is 30 bytes.

   The internal segment can have a variable length. In this case, the length of an internal segment is indirectly specified by providing the value fieldname+n in the Next Occurrence field.

To define the fields for the parent and for the containing segments, type P on the command line and press Enter. The panel shown in Figure 529 on page 669 is displayed.

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------    NOTE
Command ===>                                                   SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR3
        sequence                             Entity/Internal Segment: CREDCARD

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> ACCOUNT     Segment Level:
FLDNAME  ===> ACT_NBR                            POSITION ===> 3
BYTES    ===> 9         DATATYPE ===> Z          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> ACT_NBR
-------------    2 ------------------------------------------------------------
SEGNAME  ===> ACCOUNT     Segment Level:
FLDNAME  ===> COUNT1                             POSITION ===> 33
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>                                                                    1
-------------    3 ------------------------------------------------------------
SEGNAME  ===> ACCOUNT     Segment Level:
FLDNAME  ===> COUNT2                             POSITION ===> 35
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>                                                                    1
```

Figure 529. EKYMPPFE: Parent Segments Fields/Columns (Example 3 - PR3)

Note: The COLNAME field is blank because the count fields COUNT1 and COUNT2 are not propagated.

Type END on the command line and press Enter to return to panel EKYMP61E (shown in Figure 528 on page 668).

To define the fields for the entity segment, enter F on the command line of panel EKYMP61E and press Enter. The panel shown in Figure 530 on page 670 is displayed.

```
EKYMP62E ------- DPROP MVGIN - Internal Segment Fields/Columns ----------------
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR3
    S - Sort in POSITION sequence                Internal Segment:    CREDCARD

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
FLDNAME  ===> TYPE                               POSITION ===> 1
BYTES    ===> 10        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> TYPE
-------------    2 ------------------------------------------------------------
FLDNAME  ===> CARDID                             POSITION ===> 11
BYTES    ===> 10        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> CARDID
-------------    3 ------------------------------------------------------------
FLDNAME  ===> EXPIRES                            POSITION ===> 21
BYTES    ===> 10        DATATYPE ===> A          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> EXPIRES
```

Figure 530. EKYMP62E: Internal Segment Fields/Columns (Example 3 - PR3)

Type END on the command line and press Enter to return to panel EKYMP61E (shown in Figure 528 on page 668). Repeat the procedure to return to panel EKYMP60E (shown in Figure 527 on page 667) with the state of option 1 changed to `processed`.

Select option 2 to define the start segment. The panel shown in Figure 531 on page 671 is displayed.

```
EKYMP63E --------------- DPROP MVGIN - Start Segments -------------------------
Command ===>                                                  SCROLL ===> HALF

        Containing Segment:  ACCOUNT             Propagation Request: PR3
        Internal Segment:    CREDCARD

   Enter the definitions for one or more start segments.  To delete a
   start segment, blank out the Segment Name field.

Press <ENTER> to validate your input.  Enter END command to return.

--------------------------------------------------------------------------------
  Segment Name              ===> CREDIT    ( a name up to 8 bytes            )
  Segment Format            ===> FI        ( FI or VI                        )
  Occurrences (n or field)  ===> COUNT1
  Start                     ===> 37
  Segment Length            ===> 15        ( from 1 up to 32760              )
  Next ( field+n )          ===>
--------------------------------------------------------------------------------
  Segment Name              ===>           ( a name up to 8 bytes            )
  Segment Format            ===>           ( FI or VI                        )
  Occurrences (n or field)  ===>
  Start                     ===>
  Segment Length            ===>           ( from 1 up to 32760              )
```

Figure 531. EKYMP63E: Start Segments (Example 3 - PR3)

Type END and press Enter to return to panel EKYMP60E (shown in Figure 527 on page 667) with the state of options 1 and 2 changed to `processed`.

Type END and press Enter to return to panel EKYMP15E (shown in Figure 524 on page 666) with the state of options 1, 2, and 3 changed to `processed`.

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. The panel shown in Figure 532 is overlaid on panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

        _____DPROP MVGIN - Generate MVG Input Tables Insert Job_______    ame)
       | EKYMPIIE ==>                                                |    2
       |                                                             |
       |                                 DPROP-System:        (name) |    s
   1   |                                 Propagation Request: PR3    |    d *
   2   |                                                             |    d *
   3   |  Enter the member name to be used to store the generated    |    d *
       |  MVG input tables insert job.  The job is stored in data set |    d *
   4   |  "(library)".                                               |    d *
       |                                                             |
   5   |     Member Name    ===> PR2                                 |
       |                                                             |
   S   | Press <ENTER> to continue or enter END command to return.   |
       |                                                             |
   E   |                                                             |
       |_____________________________________________________________|


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 532. Generate MVG Input Tables Insert Job

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Providing MVGU control statements: 3

Figure 533 describes the IMS DPROP MVGU control statement that tells the MVGU to create PR1, PR2 and PR3 in the IMS DPROP directory based on data stored in the MVG input tables.

```
CREATE PR=PR1
CREATE PR=PR2      ;
CREATE PR=PR3      ;
```

Figure 533. MVGU control statements to Create the Propagation Request (Example 3)

Defining Propagation Requests in the MVG Input Tables with SQL Insert Statements: 3

Typically, propagation requests are described in the MVG input tables by a program you have provided that has a dictionary as input. It is unusual to describe propagation requests in the MVG input tables with SPUFI statements.

Figure 534 on page 673 describes SPUFI statements that define PR1 in the MVG input tables. Then, Figure 535 on page 675 and Figure 536 on page 678 describe the SPUFI statements that define PR2 and PR3 in the MVG input tables.

As shown in Figure 534 on page 673, the following rows are inserted into the MVG input tables to describe PR1:

- One DPRIPR row containing general information about PR1, which belongs to mapping case 1.
- One DPRISEG row describing the entity segment ACCOUNT propagated by PR1.

  Because DPRISEG is the root segment, there are no physical ancestors or parents to be described with other DPRISEG rows.
- One DPRITAB row identifying the target table, ACCOUNT.
- One DPRIFLD row for each of the two propagated fields, ACT_NBR and NAME.
- Two DPRIFLD rows for the two count fields of the internal segments, COUNT1 and COUNT2. The count fields contain the number of occurrences of the repeating group of fields within the IMS segment. Because count fields are not propagated, the values of the COLNAME column in the two DPRISEG rows are blank.

```
INSERT INTO T096606.DPRIPR
             ( PRID    , USERID  , PRTYPE, MAPCASE, MAPDIR,
               ERROPT  , ACTION)
      VALUES ('PR1'    ,'T096606','L'    ,'1'     ,'HR',
              'BACKOUT','REPL')                              ;

INSERT INTO T096606.DPRISEG
             ( PRID     , DBNAME  , SEGNAME ,   ROLE )
      VALUES ('PR1'     ,'DB678'  ,'ACCOUNT',  'E'  );

INSERT INTO T096606.DPRITAB
             ( PRID,   TABQUAL ,  TABNAME )
      VALUES ('PR1',  'T096606',  'ACCOUNT')                  ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR1'    ,'DB678'  , 'ACCOUNT','ACT_NBR',
              'T096606','ACCOUNT',           'ACT_NBR',
              'Z '     , 3       , 9)                        ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR1'    ,'DB678'  ,'ACCOUNT' ,'NAME'  ,
              'T096606','ACCOUNT',           'NAME'  ,
              'C '     , 12      ,  21)                      ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR1'    ,'DB678'  , 'ACCOUNT','COUNT1' ,
              'T096606','ACCOUNT',           '    '  ,
              'H '     , 33      ,  2 )                      ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR1'    ,'DB678'  , 'ACCOUNT','COUNT2' ,
              'T096606','ACCOUNT',           '    '  ,
              'H '     , 35      ,  2 )                      ;

COMMIT;
```

Figure 534. SPUFI Statements to Define PR1 in the MVG Input Tables (Example 3 - PR1)

As shown in Figure 535 on page 675, the following rows are inserted into the MVG input tables to describe PR2:

- One DPRIPR row containing general information about PR2, which belongs to mapping case 3.
- One DPRISEG row describing the containing IMS segment, ACCOUNT.

  Because ACCOUNT is the containing segment, the value of the ROLE column is set to C.

  Because DPRISEG is the root segment, there are no physical ancestors or parents to be described with other DPRISEG rows.

- One DPRISEG row describing the internal segment CREDIT (describing the repeating group of fields) being propagated by the mapping case 3 propagation request.

  Note the following column values for this DPRISEG row:

  – The ROLE column is E, identifying the segment as the entity segment being propagated by the mapping case 3 propagation request.
  – The FORMAT column is FI, identifying the segment as a fixed-length internal segment.
  – The OCCURS column is COUNT1. The internal segment has a variable number of occurrences, and the number of occurrences is stored in the field COUNT1.

    If the number of occurrences is fixed, an occurrence field is not needed; in this case, the fixed number of occurrences must be provided as a value of the OCCURS column.
  – The START column is 37, indicating that the first occurrence of the internal segment has a fixed start position within the containing IMS segment, the 37th byte.

    The first occurrence of an internal segment type could have a variable start-position. In this case, the value of the START column is coded either as *fieldname+n* or as *segname+n*.
  – The BYTES column is 15. This tells IMS DPROP that the fixed length of the internal segment is 15 bytes.

    The internal segment can have a variable length. In this case, the length of an internal segment is indirectly specified by *fieldname+n* in the NEXT column of the DPRISEG row.
- One DPRITAB row identifying the target table, CREDIT.
- One DPRIFLD row for each of the propagated fields: ACT_NBR, TYPE, AMOUNT, and LIMIT.

  For the ACT_NBR field, the value of the SEGNAME column is ACCOUNT (because this is a field located in the containing segment ACCOUNT). For the TYPE, AMOUNT, and LIMIT fields, the values of the SEGNAME column is CREDIT (because these are fields located in the internal segment CREDIT).
- One DPRIFLD row for the count field COUNT1.

  The value of the SEGNAME column is ACCOUNT (because the count field is located in the containing segment ACCOUNT).

  The value of the COLNAME column is blank (because the count field is not propagated).

```
INSERT INTO T096606.DPRIPR
            ( PRID    , USERID  , PRTYPE, MAPCASE, MAPDIR,
              ERROPT  , ACTION)
      VALUES ('PR2'    ,'T096606','L'    ,'3'     ,'HR'   ,
              'BACKOUT','REPL')                            ;

INSERT INTO T096606.DPRISEG
            ( PRID   ,  DBNAME  , SEGNAME ,   ROLE )
      VALUES ('PR2'   , 'DB678'  ,'ACCOUNT',   'C'  );


INSERT INTO T096606.DPRISEG
            ( PRID   ,  DBNAME  , SEGNAME  ,  ROLE ,
              FORMAT ,  OCCURS  , START    ,  BYTES)
      VALUES ('PR2'   , 'DB678'  ,'CREDIT ' , 'E' ,
              'FI'    , 'COUNT1' , '37'     ,  15)          ;

INSERT INTO T096606.DPRITAB
            ( PRID,   TABQUAL ,   TABNAME )
      VALUES ('PR2',  'T096606',  'CREDIT' )                ;

INSERT INTO T096606.DPRIFLD
            ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
              TABQUAL , TABNAME ,            COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR2'    ,'DB678'  , 'ACCOUNT','ACT_NBR',
              'T096606','CREDIT ',           'ACT_NBR',
              'Z '     , 3       , 9)                      ;

INSERT INTO T096606.DPRIFLD
            ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
              TABQUAL , TABNAME ,            COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR2'    ,'DB678'  , 'ACCOUNT','COUNT1',
              'T096606','CREDIT ',           '    '  ,
              'H '     , 33      ,  2 )                    ;

INSERT INTO T096606.DPRIFLD
            ( PRID    , DBNAME  , SEGNAME  , FLDNAME  ,
              TABQUAL , TABNAME ,            COLNAME  ,
              DATATYPE, POSITION, BYTES    , SCALE)
      VALUES ('PR2'    ,'DB678'  ,'CREDIT'   ,'TYPE'   ,
              'T096606','CREDIT ',           'TYPE'    ,
              'P '     , 1       , 1        , 0    )         ;

INSERT INTO T096606.DPRIFLD
            ( PRID    , DBNAME  , SEGNAME  , FLDNAME  ,
              TABQUAL , TABNAME ,            COLNAME  ,
              DATATYPE, POSITION, BYTES    , SCALE)
      VALUES ('PR2'    ,'DB678'  , 'CREDIT ','AMOUNT ' ,
              'T096606','CREDIT ',           'AMOUNT'  ,
              'P '     , 2       ,  7       , 2    )          ;

INSERT INTO T096606.DPRIFLD
            ( PRID    , DBNAME  , SEGNAME  , FLDNAME  ,
              TABQUAL , TABNAME ,            COLNAME  ,
              DATATYPE, POSITION, BYTES    , SCALE)
      VALUES ('PR2'    ,'DB678'  , 'CREDIT ','LIMIT  ' ,
              'T096606','CREDIT ',           'LIMIT '  ,
              'P '     , 9       ,  7       , 2    )          ;

COMMIT;
```

Figure 535. SPUFI Statements to Define PR2 in the MVG Input Tables (Example 3 - PR2)

As shown in Figure 536 on page 678, the following rows are inserted into the MVG input tables to describe PR3:

- One DPRIPR row containing general information about PR3, which belongs to mapping case 3.
- One DPRISEG row describing the containing IMS segment ACCOUNT.

  Because ACCOUNT is the containing segment, the value of the ROLE column is set to C.

  Because DPRISEG is the root segment, there are no physical ancestors or parents to be described with other DPRISEG rows.
- One DPRISEG row describing the internal segment CREDCARD (describing the repeating group of fields) being propagated by the mapping case 3 propagation request.

  Note the following column values for this DPRISEG row:

  – The ROLE column is E. The segment is the entity segment being propagated by the mapping case 3 propagation request.

  – The FORMAT column is FI. The segment is a fixed-length internal segment.

  – The OCCURS column is COUNT2. The internal segment has a variable number of occurrences, and the number of occurrences is stored in the field COUNT2.

    If the number of occurrences is fixed, there is no need for an occurrence field; in this case, the fixed number of occurrences must be provided as a value of the OCCURS column.

  – The START column is CREDIT+1. The first occurrence of the internal segment CREDCARD has a variable start position within the containing IMS segment. CREDIT+1 specifies that the start position is located immediately after the end of the last occurrence of the CREDIT internal segment/repeating group of fields.

  – The BYTES column is 30. The fixed length of the internal segment is 30 bytes.

    The internal segment can have a variable length. In this case, the length of an internal segment is indirectly specified by providing the value fieldname+n in the NEXT column of the DPRISEG row.
- One DPRISEG row describing the internal segment CREDIT.

  The internal segment being propagated by this Propagation Request is CREDCARD, not CREDIT. The internal segment CREDIT must nevertheless be described to IMS DPROP, because the end of its last occurrence is used to determine the start position of the first CREDCARD segment.

  The ROLE column of this DPRISEG row has the value S, indicating that this segment is a start segment. It is used to determine the start position of the first occurrence of the internal entity segment.

  The values of the other columns of this DPRISEG row are the same as in the definition of PR2.
- One DPRITAB row identifying the target table, CREDCARD.
- One DPRIFLD row for each of the propagated fields: ACT_NBR, TYPE, CARDID, and EXPIRES.

  For the ACT_NBR field, the value of the SEGNAME column is ACCOUNT (because this field is located in the containing segment ACCOUNT). For the TYPE, CARDID, and EXPIRES fields, the values of the SEGNAME column is CREDCARD (because these fields are located in the internal segment CREDCARD).

- One DPRIFLD row for the count field COUNT1, and another DPRIFLD row for the count field COUNT2.

  The value of the SEGNAME column is ACCOUNT (because the count fields are located in the containing segment ACCOUNT).

  The value of the COLNAME column is blank (because the count fields are not propagated).

```
                                                                    NOTE
INSERT INTO T096606.DPRIPR
             ( PRID    , USERID  , PRTYPE, MAPCASE, MAPDIR,
               ERROPT  , ACTION)
      VALUES ('PR3'    ,'T096606','L'    ,'3'     ,'HR'   ,
              'BACKOUT','REPL')                            ;

INSERT INTO T096606.DPRISEG
             ( PRID   ,  DBNAME  , SEGNAME ,   ROLE )
      VALUES ('PR3'   , 'DB678'  ,'ACCOUNT',   'C'  );


INSERT INTO T096606.DPRISEG
             ( PRID   ,  DBNAME  , SEGNAME  ,  ROLE ,
               FORMAT ,  OCCURS  , START    ,  BYTES)
      VALUES ('PR3'   , 'DB678'  ,'CREDIT ' ,  'S' ,
              'FI'    , 'COUNT1' , '37'     ,  15)           ;

INSERT INTO T096606.DPRISEG
             ( PRID   ,  DBNAME  , SEGNAME  ,  ROLE ,
               FORMAT ,  OCCURS  , START    ,  BYTES)
      VALUES ('PR3'   , 'DB678'  ,'CREDCARD' , 'E' ,
              'FI'    , 'COUNT2' , 'CREDIT+1',  30)           ;

INSERT INTO T096606.DPRITAB
             ( PRID,   TABQUAL ,   TABNAME )
      VALUES ('PR3',  'T096606',  'CREDCARD' )                 ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR3'    ,'DB678'  , 'ACCOUNT','ACT_NBR',
              'T096606','CREDCARD',          'ACT_NBR',
              'Z '     , 3       , 9)                        ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR3'    ,'DB678'  , 'ACCOUNT','COUNT1' ,
              'T096606','CREDCARD',          '    '  ,
              'H '     , 33      ,  2 )                      ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR3'    ,'DB678'  , 'ACCOUNT','COUNT2' ,
              'T096606','CREDCARD',          '    '  ,
              'H '     , 35      ,  2 )                      ;
```

Figure 536. SPUFI Statements to Define PR3 in the MVG Input Tables (Example 3 - PR3) (Part 1 of 2)

```
INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR3'    ,'DB678'  , 'CREDCARD','TYPE' ,
              'T096606','CREDCARD',           'TYPE' ,
              'C '     , 1       ,  10)                    ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR3'    ,'DB678'  , 'CREDCARD','CARDID'   ,
              'T096606','CREDCARD',           'CARDID'   ,
              'C '     , 11      ,  10)                    ;

INSERT INTO T096606.DPRIFLD
             ( PRID    , DBNAME  , SEGNAME  , FLDNAME,
               TABQUAL , TABNAME ,            COLNAME,
               DATATYPE, POSITION, BYTES)
      VALUES ('PR3'    ,'DB678'  , 'CREDCARD','EXPIRES' ,
              'T096606','CREDCARD',           'EXPIRES' ,
              'A '     , 21      ,  10)                    ;
COMMIT;
```

Figure 536. SPUFI Statements to Define PR3 in the MVG Input Tables (Example 3 - PR3) (Part 2 of 2)

Providing MVGU control statements: 3

Figure 537 describes the IMS DPROP MVGU control statement that tells the MVGU to create PR1, PR2, and PR3 in the IMS DPROP directory based on data stored in the MVG input tables.

```
 CREATE PR=PR1
  CREATE PR=PR2      ;
  CREATE PR=PR3      ;
```

Figure 537. MVGU control statements to Create the Propagation Requests (Example 3)

Example 4: Propagation with a WHERE Clause and PATH data: 4

This example illustrates propagation with a WHERE clause and with PATH data of a TYPE=E propagation request.

Figure 450 on page 604describes the structure of the propagated IMS database.

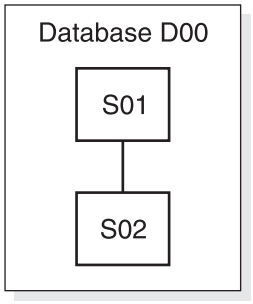

Figure 538. Structure of the IMS Database for Example 4

Overview of the Propagation Performed: 4

In this example, the propagated IMS segment S02 contains redefined data. The type of data contained in a particular occurrence of the segment is identified by the value of the field F02TYP, which in this example can have the values of A or B.

Two propagation requests are defined. PR02A propagates to table T02A those occurrences of segment S02 having the value A in F02TYP. PR02B propagates to table T02B those occurrences of segment S02 having the value B in F02TYP.

Both PR02A and PR02B include PATH data (with the PATH=ID option) in their propagation because:
- The IMS key field of the parent segment S01 is not unique
- Consequently, the IMS fully concatenated key of S02 is not unique.

Therefore, as described by the key rule 3 for extended function PRs (see the appropriate Administrators Guide for your propagation mode), a unique conceptual fully concatenated key is required for S02. In this example, a conceptual fully concatenated key can be built for S02 by combining the following fields:
- The nonunique IMS key field of S01
- An ID field of S01
- The unique IMS key field of S02

The ID field of S01 is included as PATH data in the propagation performed by PR02A and PR02B.

Figure 539 on page 681 illustrates the propagation of segment S02 (which contains redefined data) by PR02A and PR02B to the tables T02A and T02B.

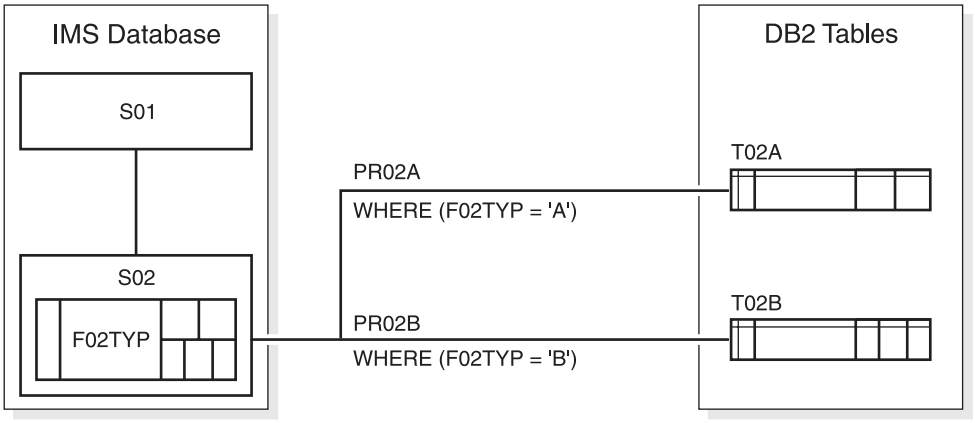

Figure 539. Overview of Propagation of Segment S02 (Example 4)

Figure 540 illustrates, for each propagation request, the mapping between the keys (and ID fields) of the segments and the DB2 primary keys of the tables.

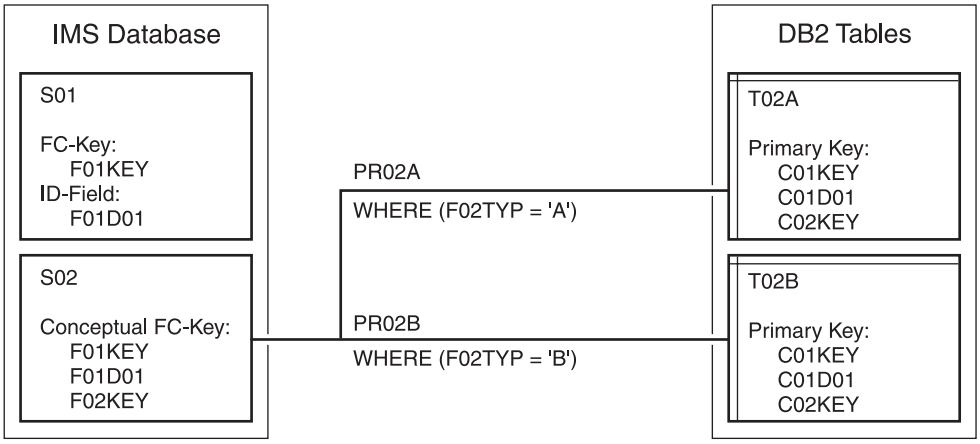

Figure 540. Key Mapping Between the Entity Segments and the Target Tables (Example 4)

Notes:

1. The IMS key field F01KEY of the parent segment S01 does not have unique values. However, combining the ID field F01D01 and F01KEY allows unique identification the segment S01.
2. The IMS fully concatenated key of the entity segment S02 does not have unique values. However, combining the following fields allows a conceptual fully concatenated key to be built with unique values:
   - F01KEY (the IMS key field of segment S01)
   - F01D01 (an ID field of segment S01)
   - F02KEY (the IMS key field of segment S02)
3. It is this conceptual fully concatenated key of segment S02 that is mapped to the DB2 primary key of the tables T02A and T02B.

   Mapping the field F01D01 of the parent segment implies that F01D01 is included as PATH data in the mapping performed by PR02A and PR02B.

Field Mapping Overview: 4

Table 50 and Table 51 show, for each propagation request, the characteristics of the source IMS fields and the target DB2 columns. The tables contains the following information:

- For the IMS source field:
  - The name of the IMS field (as defined to IMS DPROP either on the NAME keyword of the DataRefresher FIELD statement, or in the FLDNAME column of the DPRIFLD MVG input table)
  - The data type (as defined to IMS DPROP either on the TYPE keyword of the DataRefresher FIELD statement, or in the DATATYPE column of the DPRIFLD MVG input table)
  - The length in bytes (as defined to IMS DPROP either on the BYTES keyword of the DataRefresher FIELD statement or in the BYTES column of the DPRIFLD MVG input table)
- For the DB2 target column:
  - The name of the target DB2 column (as defined to DB2 in the CREATE TABLE statement)
  - The data type of the target DB2 column (as defined to DB2 in the CREATE TABLE statement)

Table 50. Source Fields and Target Columns for PR02A

Source IMS fields				Target DB2 columns	
Field Name	**See Note Below**	**Data Type**	**Length (in bytes)**	**Column Name**	**Data Type**
F01KEY	1	C	5	C01KEY	char(5)
F01D01	2	C	10	C01D01	char(10)
F02KEY	1	C	6	C02KEY	char(6)
F02TYP	3	C	1	C02TYP	char(1)
F02AD1		C	5	C02AD1	char(5)
F02AD2		C	5	C02AD2	char(5)

Table 51. Source Fields and Target Columns for PR02B

Source IMS fields				Target DB2 columns	
Field Name	**See Note Below**	**Data Type**	**Length (in bytes)**	**Column Name**	**Data Type**
F01KEY	1	C	5	C01KEY	char(5)
F01D01	2	C	10	C01D01	char(10)
F02KEY	1	C	6	C02KEY	char(6)
F02TYP	3	C	1	C02TYP	char(1)
F02BD1		C	2	C02BD1	char(2)
F02BD2		C	4	C02BD2	char(4)
F02BD3		C	10	C02BD3	char(10)

Notes:

1. F01KEY, and F02KEY are the key fields of the IMS segments S01 and S02. They are propagated to columns of the DB2 primary key.

2. F01D01 is an ID field of segment S01. The combination of F01KEY and F01D01 uniquely identify each occurrence of segment S01.

   F01D01 is propagated as PATH data to a column of the DB2 primary key.
3. F02TYP is a field of S02 describing which redefined data is contained in a particular occurrence of segment S02. F02TYP is therefore used in the WHERE clause of the propagation request.

   As required by IMS DPROP rules for the WHERE clause of TYPE=E propagation requests. fields included in the WHERE clause must be propagated to a DB2 column of the target table.

DBD Definitions: 4

The DBD control statements for the model database are shown in Figure 541.

```
         DBD NAME=D00,VERSION=V123456789,                              C
               EXIT=(EKYRUP00,PATH,(CASCADE,PATH)),                    C
               ACCESS=(HDAM,OSAM),RMNAME=(DFSHDC40,5,4)
         DATASET DD1=HDAM,SIZE=4096,DEVICE=3380
*
         SEGM  NAME=S01,PARENT=0,BYTES=60
         FIELD  NAME=(F01KEY,SEQ,M),BYTES=5,START=1
         FIELD  NAME=F01D01,BYTES=10,START=6
*
         SEGM  NAME=S02,PARENT=S01,BYTES=(126,26)
         FIELD  NAME=(F02KEY,SEQ,U),BYTES=6,START=3
*
         DBDGEN
         FINISH
         END
```

Figure 541. DBD control statements for the Model Database (Example 4)

The EXIT keyword in the DBD control statement tells IMS to give control to EKYRUP00, the IMS DPROP-provided DL/I Data Capture Exit routine, when a DL/I call updates a segment of the database DB678.

The definition of the EXIT= keyword value specifies PATH,(CASCADE,PATH). This is because the PR propagates PATH data. Consequently, the EXIT= keyword value must specify, as described in the appropriate Administrators Guide for your propagation mode:
- The PATH data option
- The PATH suboption of CASCADE

The field F01KEY of segment S01 is defined to IMS with the SEQ,M specification. This means that F01KEY is a key field that does not have unique values.

IMS DPROP rules for a TYPE=E PR require that ID fields be defined in the IMS DBD. Therefore, the field F01D01 (the ID field of segment S01 that is included in the mapping as PATH data) is defined in the DBD. Fields other than the key fields and this ID field do not need to be defined in the IMS DBD.

Table Definitions: 4

Figure 542 describes the SQL statements to create the target tables T02A and T02B. As required by IMS DPROP, these tables are defined with a DB2 primary key.

```
CREATE TABLE T02A
            (C01KEY   CHAR(5)           NOT NULL,
             C01D01   CHAR(10)          NOT NULL,
             C02KEY   CHAR(6)           NOT NULL,
             C02TYP   CHAR(1)           NOT NULL,
             C02AD1   CHAR(5)           NOT NULL WITH DEFAULT,
             C02AD2   CHAR(5)           NOT NULL WITH DEFAULT,
          PRIMARY KEY (C01KEY, C01D01, C02KEY))
       IN DU096606.PROPTSA   ;

CREATE UNIQUE INDEX XN02A ON T02A (C01KEY, C01D01, C02KEY)
       USING VCAT KOE ;



CREATE TABLE T02B
            (C01KEY   CHAR(5)           NOT NULL,
             C01D01   CHAR(10)          NOT NULL,
             C02KEY   CHAR(6)           NOT NULL,
             C02TYP   CHAR(1)           NOT NULL,
             C02BD1   CHAR(2)           NOT NULL WITH DEFAULT,
             C02BD2   CHAR(4)           NOT NULL WITH DEFAULT,
             C02BD3   CHAR(10)          NOT NULL WITH DEFAULT,
          PRIMARY KEY (C01KEY, C01D01, C02KEY))
       IN DU096606.PROPTSB   ;

CREATE UNIQUE INDEX XN02B ON T02B (C01KEY, C01D01, C02KEY)
       USING VCAT KOE ;
```

Figure 542. Tables T02A and T02B (Example 4)

Defining Propagation Requests with DataRefresher: 4

Defining the DXTPSB: 4

DataRefresher requires that you describe the IMS database to DataRefresher by providing a CREATE DXTPSB command. The DataRefresher control statements for our model database are shown in Figure 543 on page 685.

The control statements, which are highlighted numerically (**1** for example), are further described following the figure.

```
                                                                    NOTE

CREATE DXTPSB    NAME=PSB00

       DXTPCB    NAME=PCB001,DBACCESS=HDAM,DBNAME=D00

         SEGMENT NAME=S01,
                 PARENT=0,BYTES=60

             FIELD    NAME=F01KEY , START=1,  BYTES=5 ,  SEQFLD=R
             FIELD    NAME=F01D01 , START=6,  BYTES=10 , TYPE=C


         SEGMENT NAME=S02,                                          1
                 PARENT=S01,BYTES=126,FORMAT=V

             FIELD    NAME=F02KEY , START=3,  BYTES=6 ,  SEQFLD=R
             FIELD    NAME=F02TYP , START=9,  BYTES=1 ,  TYPE=C

             FIELD    NAME=F02AD1 , START=10, BYTES=5 ,  TYPE=C
             FIELD    NAME=F02AD2 , START=15, BYTES=5 ,  TYPE=C

             FIELD    NAME=F02BD1 , START=10, BYTES=2 ,  TYPE=C
             FIELD    NAME=F02BD2 , START=12, BYTES=4 ,  TYPE=C
             FIELD    NAME=F02BD3 , START=16, BYTES=10,  TYPE=C ;
```

Figure 543. DataRefresher control statements for the Model Database (Example 4)

Notes:

1. The SEGMENT statement with the name S02 defines the IMS segment to DataRefresher. FORMAT=V indicates that the IMS segment has a variable length.

   The segment name defined to DataRefresher must match the segment name defined to IMS in the DBD.

   The segment S02 has the following group of fields:
   - The key field F02KEY
   - The field F02TYP, describing which type of data is contained in a particular occurrence of the segment
   - The fields F02AD1 and F02AD2 (if F02TYP has the value A)
   - The fields F02BD1, F02BD2, and F02BD3 (if F02TYP has the value B). These fields overlay F02AD1 and F02AD2.

Defining the DXTVIEW: 4

DataRefresher requires a view description for each hierarchical path of the database. The DataRefresher command used to define the view for our example is shown in Figure 544.

```
                                                        NOTE
   CREATE
      DXTVIEW NAME     = VIEW02,
              DXTPSB   = PSB00,                         1
              DXTPCB   = PCB001,                        2
              SEGMENT  = S02,
              MINSEGM  = S02,
              FIELDS   = *          ;                   3
```

Figure 544. CREATE DXTVIEW (Example 4)

Notes:

1. The DXTPSB=KOEPSB2 keyword identifies the previously created DXTPSB.
2. The DXTPCB=PCB001 keyword identifies the DXTPCB of the DXTPSB that describes the propagated IMS database.
3. The FIELDS=* operand tells DataRefresher that all fields of the described hierarchical path should be included in the DXTVIEW.

Providing DataRefresher UIM SUBMIT Commands and EXTRACT Statements: 4

Figure 545 on page 687 illustrates the DataRefresher SUBMIT commands and EXTRACT statements used to define the propagation requests that will propagate the segment S02.

The DataRefresher SUBMIT/EXTRACT creates the propagation requests in the IMS DPROP directory but does not load the data to the DB2 tables. This is the responsibility of the DataRefresher DEM. Refer to “Providing DataRefresher DEM control statements: 4 ” on page 688 for examples of a DataRefresher DEM control statement.

```
                                                                    NOTE

SUBMIT   EXTID=PR02A,
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,MAPDIR=HR,MAPCASE=1,PATH=ID,       1
                         ACTION=REPL,ERROPT=BACKOUT',
               FORMAT=SOURCE,
               CD=JCS,
               JCS=DDJCS01

EXTRACT
    INTO T02A    (C01KEY NOT NULL,
                  C01D01 NOT NULL,
                  C02KEY NOT NULL,
                  C02TYP NOT NULL,
                  C02AD1 NOT NULL WITH DEFAULT,
                  C02AD2 NOT NULL WITH DEFAULT
                 )
    OPTIONS      (FLDERR(HALT))
    SELECT        F01KEY,
                  F01D01,
                  F02KEY,
                  F02TYP,                                            2
                  F02AD1,
                  F02AD2
                 FROM VIEW02                                         3
    WHERE        (F02TYP = 'A') ;                                    4


SUBMIT   EXTID=PR02B,
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,MAPDIR=HR,MAPCASE=1,PATH=ID,       1
                         ACTION=REPL,ERROPT=BACKOUT',
               FORMAT=SOURCE,
               CD=JCS,
               JCS=DDJCS01

EXTRACT
    INTO T02B    (C01KEY NOT NULL,
                  C01D01 NOT NULL,
                  C02KEY NOT NULL,
                  C02TYP NOT NULL,
                  C02BD1 NOT NULL WITH DEFAULT,
                  C02BD2 NOT NULL WITH DEFAULT,
                  C02BD3 NOT NULL WITH DEFAULT
                 )
    OPTIONS      (FLDERR(HALT))
    SELECT        F01KEY,
                  F01D01,
                  F02KEY,
                  F02TYP,                                            2
                  F02BD1,
                  F02BD2,
                  F02BD3
                 FROM VIEW02                                         3
    WHERE        (F02TYP = 'B') ;                                    4
```

Figure 545. DataRefresher SUBMIT/EXTRACT for PR02A and PR02B (Example 4)

Notes:

1. PATH=ID must be specified, because the propagation requests include PATH data in their mapping.

2. F02TYP is the field used in the WHERE clause of the propagation request. It must be included in the propagation, because IMS DPROP requires this for the WHERE clause of TYPE=E propagation requests.
3. VIEW02 is the previously created DXTVIEW.
4. This is the WHERE clause of the propagation request definition.

Providing DataRefresher DEM control statements: 4

Figure 546 shows an example of a DataRefresher DEM control statement that you can use to perform the DataRefresher extract for all propagation requests using PSB00 as DXTPSB.

```
INITDEM   NAME=DEM01
USE DXTPSB=PSB00;
```

Figure 546. DataRefresher DEM control statement to Perform an Extract (Example 4)

Defining Propagation Requests with the MVG Input Tables ISPF Application: 4

In this example, two different propagation requests are defined on the MVG input tables. The entire panel flow is shown for each propagation request.

Call the MVGIN Front End Application as described in Chapter 20, “CCU Front End ,” on page 523. The panel shown in Figure 547 is displayed:

```
EKYMP00E ------------- DPROP MVG Input Tables Application ---------------------
Command ===>

                                                          DPROP Environ-
   1 - Insert, Update, Show, Delete a Propagation Request  ment  : (name)
   2 - Propagation Request Overview to Update, Show,       DPROP-
       Delete a Propagation Request                        System: (name)
   3 - Setup IMS and DPROP System Environment              DPROP-System-
   4 - Edit existing MVG Input Tables Insert/Delete Jobs   Number: (number)

   E - Return


     Enter the fully qualified name of the partitioned data set to be
     used to store the generated MVGIN job control:
        Data Set Name ===>
     Enter the volume serial name to be used to allocate a new data set:
        Volume Serial ===>


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 547. EKYMP00E: DPROP MVG Input Tables Application (Example 4)

The first time you call the MVGIN Front End Application, you have to complete the setup panels and enter a data set name in the Data Set Name field.

Panel Flow for PR02A: 4

The panel shown in Figure 548 on page 689 is displayed.

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                        DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR02A


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                 ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Figure 548. EKYMP10E: Maintain a Propagation Request (Example 4 - PR02A)

Type I in the Option field and PR02A in the propagation request Name field and press Enter.

After your input is validated, the panel shown in Figure 549 is displayed.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR02A

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 549. EKYMP15E: Propagation Request Definition Selection (Example 4 - PR02A)

Select option 1 to enter the general propagation request specifications. The panel shown in Figure 550 on page 690 is displayed.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------    NOTE
Command ===>
                                                Propagation Request: PR02A

                                                                      More:   - +
  PR Type                   ===> E         ( E, L, F or U                   )
  Mapping Case              ===> 1         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===> ID        ( ID, DENORM or blank            )      1
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <
  User ID                   ===> T096606   ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 550. EKYMP20E: Propagation Request (Example 4 - PR02A)

Note: The Type of Path Data field must contain the value ID because the propagation request includes path data in its mapping and is defined as propagation request Type=E.

Type END and press Enter to return to panel EKYMP15E (shown in Figure 549 on page 689) with the state of option 1 changed to `processed`.

Select option 2 on panel EKYMP15E to define the DB2 table. The panel shown in Figure 551 on page 691 is displayed.

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                   Propagation Request: PR02A

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===> T096606     ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> T02A                ( a name up to 18 bytes )



         Note: The DB2 table you enter is not checked for existence.





Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 551. EKYMP30E: DB2 Table (Example 4 - PR02A)

Type END and press Enter to return to panel EKYMP15E (shown in Figure 549 on page 689) with the state of options 1 and 2 changed to `processed`.

Select option 3 on panel EKYMP15E to define the segments, fields, and columns. The panel shown in Figure 552 is displayed.

```
EKYMP40E --- DPROP MVGIN - Entity Segment for a Mapping Case 1 PR -------------
Command ===>

    F - Fields for the entity segment            Propagation Request: PR02A
    P - Fields for the parent segments           Database-
                                                 Organization: HDAM/OSAM

  IMS Database Name         ===> D00       ( a name up to 8 bytes           )
  Segment Name              ===> S02       ( a name up to 8 bytes           )


  PCB Label Name            ===>           ( a name up to 8 bytes           )


  Segment Exit Routine      ===>           ( a name up to 8 bytes, optional )
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760             )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                         )


Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 552. EKYMP40E: Entity Segment for a Mapping Case 1 Propagation Request (Example 4 - PR02A)

To define the fields for the parent segments, type P on the command line and press Enter. The panel shown in Figure 553 on page 692 is displayed.

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR02A
        sequence                             Entity/Internal Segment: S02

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S01         Segment Level:
FLDNAME  ===> F01KEY                              POSITION ===> 1
BYTES    ===> 5         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C01KEY
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S01         Segment Level:
FLDNAME  ===> F01D01                              POSITION ===> 6
BYTES    ===> 10        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C01D01
```

Figure 553. EKYMPPFE: Parent Segments Fields/Columns (Example 4 - PR02A)

Type END on the command line and press Enter to return to panel EKYMP40E (shown in Figure 552 on page 691).

To define the fields for the entity segment, type F on the command line of panel EKYMP40E and press Enter. The panel shown in Figure 554 on page 693 is displayed.

```
EKYMP41E -------- DPROP MVGIN - Entity Segment Fields/Columns -----------------    NOTE
Command ===>                                                   SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR02A
    S - Sort in POSITION sequence                Entity Segment:      S02

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
FLDNAME  ===> F02KEY                             POSITION ===> 3
BYTES    ===> 6         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02KEY
-------------    2 ------------------------------------------------------------
FLDNAME  ===> F02TYP                             POSITION ===> 9                  1
BYTES    ===> 1         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02TYP
-------------    3 ------------------------------------------------------------
FLDNAME  ===> F02AD1                             POSITION ===> 10
BYTES    ===> 5         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02AD1
-------------    4 ------------------------------------------------------------
FLDNAME  ===> F02AD2                             POSITION ===> 15
BYTES    ===> 5         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02AD2
```

Figure 554. EKYMP41E: Entity Segment Fields/Columns (Example 4 - PR02A)

Note: F02TYP is the field used in the WHERE clause of the propagation request. It must be included in the propagation, because DPROP requires this for the WHERE clause of Propagation Request Type=E propagation requests.

Type END on the command line and press Enter to return to panel EKYMP40E (shown in Figure 552 on page 691).

Repeat the procedure to return to panel EKYMP15E (shown in Figure 549 on page 689) with the state of options 1, 2, and 3 changed to `processed`.

Select option 4 on panel EKYMP15E to define the WHERE clause. The panel shown in Figure 555 on page 694 is displayed.

```
ISREDDE   PREFIX.USERID.MVG.WORKDSN.WHERE.CLAUSE ------------ COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> HALF
****** **************************** TOP OF DATA ******************************
=NOTE= ===>  ENTER THE WHERE CLAUSE BETWEEN COLUMNS 1 TO 72
====== ----+---1----+----2----+----3----+----4----+----5----+----6----+----7--
000001  WHERE (F02TYP = 'A')
****** *************************** BOTTOM OF DATA ***************************
```

Figure 555. MVG Input Tables ISPF Application: WHERE CLAUSE for Example 4 - PR02A

Type END on the command line and press Enter to return to panel EKYMP15E (shown in Figure 549 on page 689) with the state of options 1, 2, 3, and 4 changed to `processed`.

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. The panel shown in Figure 556 is overlaid on panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

        _____DPROP MVGIN - Generate MVG Input Tables Insert Job_____     ame)
       | EKYMPIIE ==>                                                |  02A
       |                                                             |
       |                                DPROP-System:        (name)  |  s
   1   |                                Propagation Request: PR02A   |  d *
   2   |                                                             |  d *
   3   |  Enter the member name to be used to store the generated    |  d *
       |  MVG input tables insert job.  The job is stored in data set |  d *
   4   |  "(library)".                                               |  d *
       |                                                             |
   5   |     Member Name    ===> PR02A                               |
       |                                                             |
   S   | Press <ENTER> to continue or enter END command to return.   |
       |                                                             |
   E   |                                                             |
       |_____________________________________________________________|


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 556. Generate MVG Input Tables Insert Job

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Return to panel EKYMP10E to define Propagation Request PR02B.

Panel Flow for PR02B: 4

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                        DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR02B


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                 ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Type I in the Option field and PR02B in the propagation request Name field and press Enter.

After your input is validated, the panel shown in Figure 557 is displayed.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR02B

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 557. EKYMP15E: Propagation Request Definition Selection (Example 4 - PR02B)

Select option 1 to enter the general propagation request specifications. The panel shown in Figure 558 on page 696 is displayed.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------    NOTE
Command ===>
                                                Propagation Request: PR02B

                                                                      More:   - +
  PR Type                   ===> E         ( E, L, F or U                   )
  Mapping Case              ===> 1         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===> ID        ( ID, DENORM or blank            )      1
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <
  User ID                   ===> T096606   ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 558. EKYMP20E: Propagation Request (Example 4 - PR02B)

Note: The Type of Path Data field must contain the value ID because the propagation request includes path data in its mapping and is defined as propagation request Type=E.

Type END on the command line and press Enter to return to panel EKYMP15E (shown in Figure 557 on page 695) with the state of option 1 changed to `processed`.

Select option 2 on panel EKYMP15E to define the DB2 table. The panel shown in Figure 559 on page 697 is displayed.

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                   Propagation Request: PR02B

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===> T096606     ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> T02B                ( a name up to 18 bytes )


         Note: The DB2 table you enter is not checked for existence.







Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 559. EKYMP30E: DB2 Table (Example 4 - PR02B)

Type END on the command line and press Enter to return to panel EKYMP15E (shown in Figure 557 on page 695) with the state of options 1 and 2 changed to processed.

Select option 3 on panel EKYMP15E to define the segments, fields and Columns. The panel shown in Figure 560 is displayed.

```
EKYMP40E --- DPROP MVGIN - Entity Segment for a Mapping Case 1 PR -------------
Command ===>

    F - Fields for the entity segment            Propagation Request: PR02B
    P - Fields for the parent segments           Database-
                                                 Organization: HDAM/OSAM

  IMS Database Name         ===> D00       ( a name up to 8 bytes           )
  Segment Name              ===> S02       ( a name up to 8 bytes           )


  PCB Label Name            ===>           ( a name up to 8 bytes           )


  Segment Exit Routine      ===>           ( a name up to 8 bytes, optional )
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760             )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                         )


Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 560. EKYMP40E - Entity Segment for a Mapping Case 1 Propagation Request (PR02B)

To define the fields for the parent segments, type P on the command line and press Enter. The panel shown in Figure 561 on page 698 is displayed.

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR02B
        sequence                             Entity/Internal Segment: S02

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S01         Segment Level:
FLDNAME  ===> F01KEY                              POSITION ===> 1
BYTES    ===> 5         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C01KEY
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S01         Segment Level:
FLDNAME  ===> F01D01                              POSITION ===> 6
BYTES    ===> 10        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C01D01
```

Figure 561. EKYMPPFE: Parent Segments Fields/Columns (Example 4 - PR02B)

Type END on the command line and press Enter to return to panel EKYMP40E (shown in Figure 560 on page 697).

To define the fields for the entity segment, enter F on the command line of panel EKYMP40E and press Enter. The panel shown in Figure 562 on page 699 is displayed.

```
EKYMP41E -------- DPROP MVGIN - Entity Segment Fields/Columns -----------------    NOTE
Command ===>                                                   SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR02B
    S - Sort in POSITION sequence                Entity Segment:      S02

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
FLDNAME  ===> F02KEY                             POSITION ===> 3
BYTES    ===> 6         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02KEY
-------------    2 ------------------------------------------------------------
FLDNAME  ===> F02TYP                             POSITION ===> 9                  1
BYTES    ===> 1         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02TYP
-------------    3 ------------------------------------------------------------
FLDNAME  ===> F02BD1                             POSITION ===> 10
BYTES    ===> 2         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02BD1
-------------    4 ------------------------------------------------------------
FLDNAME  ===> F02BD2                             POSITION ===> 12
BYTES    ===> 4         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02BD2
-------------    5 ------------------------------------------------------------
FLDNAME  ===> F02BD3                             POSITION ===> 16
BYTES    ===> 10        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C02BD3
```

Figure 562. EKYMP41E: Entity Segment Fields/Columns (Example 4 - PR02B)

Note: F02TYP is the field used in the WHERE clause of the propagation reque st. It must be included in the propagation, because it is required for the WHERE of PR Type=E PRs.

Type END on the command line and press Enter to return to panel EKYMP40E (shown in Figure 560 on page 697).

Repeat the procedure to return to panel EKYMP15E (shown in Figure 557 on page 695) with the state of options 1, 2, and 3 changed to `processed`.

Select option 4 on panel EKYMP15E to define the WHERE clause. The panel shown in Figure 563 on page 700 is displayed.

```
ISREDDE   PREFIX.USERID.MVG.WORKDSN.WHERE.CLAUSE ------------ COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> HALF
****** **************************** TOP OF DATA ******************************
=NOTE= ===>  ENTER THE WHERE CLAUSE BETWEEN COLUMNS 1 TO 72
====== ----+---1----+----2----+----3----+----4----+----5----+----6----+----7--
000001  WHERE (F02TYP = 'B')
****** *************************** BOTTOM OF DATA ***************************
```

Figure 563. ISREDDE: WHERE Clause (Example 4 - PR02B)

Type END on the command line and press Enter return to panel EKYMP15E (shown in Figure 557 on page 695) with the state of options 1, 2, 3, and 4 changed to `processed`.

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. The panel shown in Figure 564 is overlaid on panel EKYNMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

        _____DPROP MVGIN - Generate MVG Input Tables Insert Job _______  ame)
       | EKYMPIIE ==>                                                  | 02B
       |                                                               |
       |                                DPROP-System:        (name)    | s
   1   |                                Propagation Request: PR02B     | d *
   2   |                                                               | d *
   3   |  Enter the member name to be used to store the generated      | d *
       |  MVG input tables insert job.  The job is stored in data set  | d *
   4   |  "(library)".                                                 | d *
       |                                                               |
   5   |     Member Name    ===> PR02B                                 |
       |                                                               |
   S   | Press <ENTER> to continue or enter END command to return.     |
       |                                                               |
   E   |                                                               |
       |_______________________________________________________________|


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 564. EKYMP15E: Propagation Request Definition Selection (Example 4 - PR02B)

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Providing MVGU control statements: 4

Figure 565 describes the IMS DPROP MVGU control statement that tells the MVGU to create PR02A and PR02B in the IMS DPROP directory, based on data stored in the MVG input tables.

```
CREATE PR=PR02A
CREATE PR=PR02B;
```

Figure 565. MVGU control statements to Create PR02A and PR02B (Example 4)

Defining Propagation Requests in the MVG Input Tables with SQL Insert Statements: 4

Typically, propagation requests are described in the MVG input tables by a program you have provided that has a dictionary as input. It is unusual to describe propagation requests in the MVG input tables with SPUFI statements.

Figure 566 on page 702 and Figure 567 on page 703 describe SPUFI statements that define PR02A and PR02B in the MVG input tables.

As shown in Figure 566 on page 702 and Figure 567 on page 703, the following rows are inserted into the MVG input tables to describe each PR:

- One DPRIPR row containing general information about the propagation request, which belongs to mapping case 1.
- One DPRISEG row:
  - For the entity segment S02
  - For the only physical parent/ancestor segment, S01
- One DPRITAB row identifying the target table, T02A/T02B
- One DPRIFLD row for each of the propagated fields
- One DPRIWHR row describing the WHERE clause of the PR

```
                                                                    NOTE
INSERT INTO T096606.DPRIPR
            ( PRID,     USERID,   PRTYPE, MAPCASE, MAPDIR, PATH,
              ERROPT,   ACTION)
      VALUES ('PR02A',  'T096606', 'E',   '1',      'HR', 'ID',         1
             'BACKOUT','REPL')                          ;

INSERT INTO T096606.DPRISEG
            ( PRID,   DBNAME, SEGNAME, ROLE )
      VALUES ('PR02A', 'D00', 'S01',   'P' )              ;

INSERT INTO T096606.DPRISEG
            ( PRID,   DBNAME, SEGNAME, ROLE )
      VALUES ('PR02A', 'D00', 'S02',   'E' )              ;

INSERT INTO T096606.DPRITAB
            ( PRID,   TABQUAL,   TABNAME )
      VALUES ('PR02A', 'T096606', 'T02A')                    ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02A',  'D00',    'S01',   'F01KEY' ,
             'T096606','T02A',   'C01KEY',
             'C ',      1,        5)                      ; INSERT INTO T096606.DPRIFLD
( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02A',  'D00',    'S01',   'F01D01' ,
             'T096606','T02A',   'C01D01',
             'C ',      6,        10)                     ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02A',  'D00',    'S02',   'F02KEY' ,
             'T096606','T02A',   'C02KEY',
             'C ',      3,        6)                      ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02A',  'D00',    'S02',   'F02TYP' ,                  2
             'T096606','T02A',   'C02TYP',
             'C ',      9,        1)                      ; INSERT INTO T096606.DPRIFLD
( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02A',  'D00',    'S02',   'F02AD1' ,
             'T096606','T02A',   'C02AD1',
             'C ',      10,       5)                      ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02A',  'D00',    'S02',   'F02AD2' ,
             'T096606','T02A',   'C02AD2',
             'C ',      15,       5)                      ;

INSERT INTO T096606.DPRIWHR
            ( PRID,   SEQUENCE,  WHRTEXT )
      VALUES ('PR02A', 1        , 'WHERE (F02TYP = ''A'')')     ;
```

Figure 566. SPUFI Statements to Define PR02A in the MVG Input Table (Example 4)

Notes:

1. The value ID must be inserted into the column PATH, because the propagation requests include PATH data in their mapping and are defined as TYPE=E propagation requests.

2. F02TYP is the field used in the WHERE clause of the propagation request. It must be included in the propagation, because IMS DPROP requires this for the WHERE clause of TYPE=E propagation requests.

```
                                                                    NOTE
INSERT INTO T096606.DPRIPR
            ( PRID,     USERID,   PRTYPE, MAPCASE, MAPDIR, PATH,
              ERROPT,   ACTION)
      VALUES ('PR02B',  'T096606', 'E',   '1',      'HR', 'ID',         1
             'BACKOUT','REPL')                          ;

INSERT INTO T096606.DPRISEG
            ( PRID,   DBNAME, SEGNAME, ROLE )
      VALUES ('PR02B', 'D00', 'S01',   'P' )              ;

INSERT INTO T096606.DPRISEG
            ( PRID,   DBNAME, SEGNAME, ROLE )
      VALUES ('PR02B', 'D00', 'S02',   'E' )              ;

INSERT INTO T096606.DPRITAB
            ( PRID,   TABQUAL,   TABNAME )
      VALUES ('PR02B', 'T096606', 'T02B')                    ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02B',  'D00',    'S01',   'F01KEY' ,
             'T096606','T02B',   'C01KEY',
             'C ',      1,        5)                      ; INSERT INTO T096606.DPRIFLD
( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02B',  'D00',    'S01',   'F01D01' ,
             'T096606','T02B',   'C01D01',
             'C ',      6,        10)                     ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02B',  'D00',    'S02',   'F02KEY' ,
             'T096606','T02B',   'C02KEY',
             'C ',      3,        6)                      ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02B',  'D00',    'S02',   'F02TYP' ,                 2
             'T096606','T02B',   'C02TYP',
             'C ',      9,        1)                      ; INSERT INTO T096606.DPRIFLD
( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02B',  'D00',    'S02',   'F02BD1' ,
             'T096606','T02B',   'C02BD1',
             'C ',      10,       2)                      ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02B',  'D00',    'S02',   'F02BD2' ,
             'T096606','T02B',   'C02BD2',
             'C ',      12,       4)                      ;

INSERT INTO T096606.DPRIFLD
            ( PRID,     DBNAME,   SEGNAME, FLDNAME,
              TABQUAL,  TABNAME,  COLNAME,
              DATATYPE, POSITION, BYTES)
      VALUES ('PR02B',  'D00',    'S02',   'F02BD3' ,
             'T096606','T02B',   'C02BD3',
             'C ',      16,       10)                     ;

INSERT INTO T096606.DPRIWHR
            ( PRID,   SEQUENCE,  WHRTEXT )
      VALUES ('PR02B', 1        , 'WHERE (F02TYP = ''B'')')     ;
```

Figure 567. SPUFI Statements to Define PR02B in the MVG Input Table (Example 4)

Notes:

1. The value ID must be inserted into the column PATH, because the propagation requests include PATH data in their mapping and are defined as TYPE=E propagation requests.
2. F02TYP is the field used in the WHERE clause of the propagation request. It must be included in the propagation, because IMS DPROP requires this for the WHERE clause of TYPE=E propagation requests.

Providing MVGU control statements: 4

Figure 568 describes the IMS DPROP MVGU control statements that tell the MVGU to create PR02A and PR02B in the IMS DPROP directory based on data stored in the MVG input tables.

```
CREATE PR=PR02A
CREATE PR=PR02B    ;
```

Figure 568. MVGU control statements to Create PR02A and PR02B (Example 4)

Example 5: Propagating a Complete IMS Database Structure

To describe different propagation possibilities, the database shown in Figure 569 serves as our model for the examples of this topic.

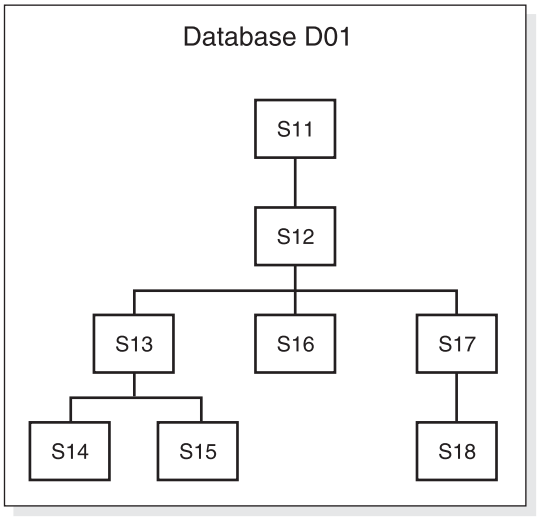

Figure 569. Model Database for Example 5

This example describes:

- How to propagate one IMS segment type to one DB2 table with propagation requests belonging to mapping case 1.

  This applies to segments S11, S12, and S16.

  – S11 and S12 have unique key fields.

  – S16 has no key field. Although the segment has no key, the content of one of the data fields can be used to uniquely identify the segment under its parent. Therefore, you can build in target table T16 a unique DB2 primary key (as required by the generalized mapping logic of IMS DPROP).

- How to propagate one IMS segment type and immediately-subordinated segment types (with a maximum of one occurrence of each segment type per parent) to one DB2 table, with propagation requests belonging to mapping case 2. This applies to segments S13, S14, S15, S17, and S18.

Visual Overview of the Performed Propagation: 5

Figure 570 illustrates the propagation of the segments of our model IMS database D01 to DB2 tables.

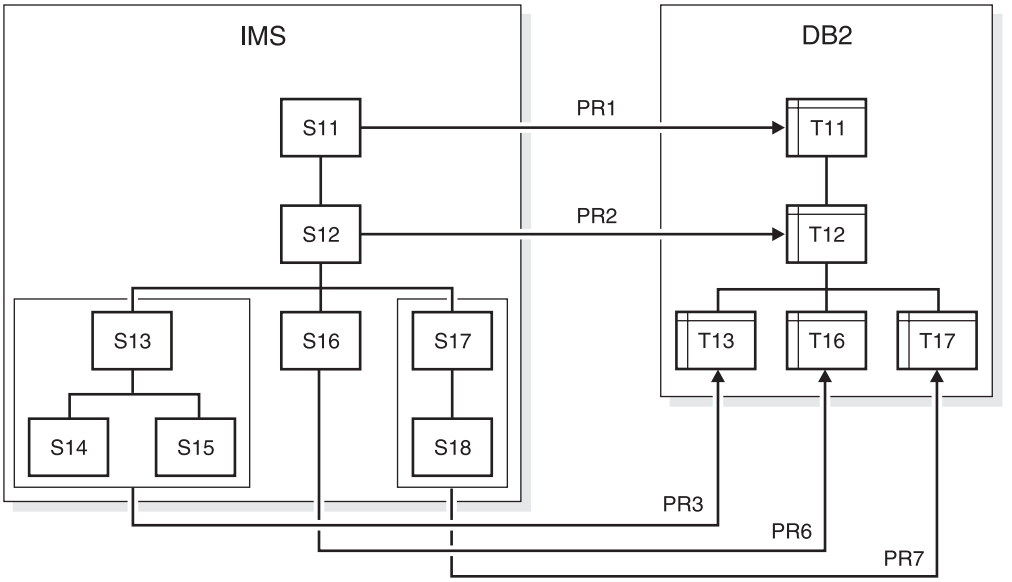

Figure 570. Overview of Propagation of Database D01

Figure 637 on page 768 illustrates that:
- Segment S11 is propagated by PR1 to table T11
- Segment S12 is propagated by PR2 to table T12
- Segments S13, S14, and S15 are propagated by PR3 to table T13
- Segment S16 is propagated by PR6 to table T16
- Segments S17 and S18 are propagated by PR7 to table T17

Figure 571 on page 706 is an overview of how the IMS fully concatenated key is mapped to the DB2 primary key. It shows:
- The implementation of DB2 RIRs between the propagated tables, which match the parent/child relationships in the IMS database.
- How IMS fields belonging to the fully concatenated key of the entity segments are mapped to columns of the DB2 primary key.
- How IMS fields belonging to the concatenated key of the propagated entity segment are mapped to columns of the DB2 foreign key.

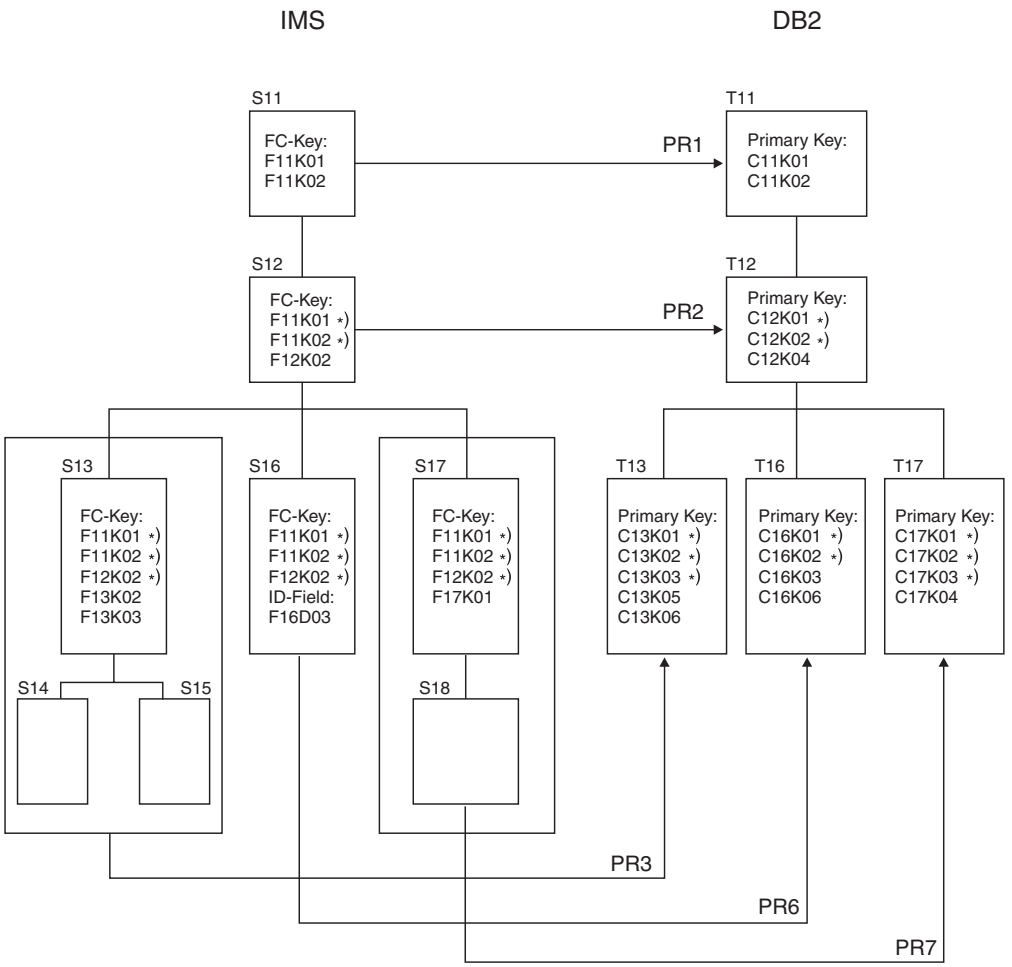

Figure 571. Mapping of the Keys

Figure 571 shows:

- For each entity segment:
  - The fields that belong to the fully concatenated key. For example, the following fields of entity segment S13 belong to its fully concatenated key: F11K01, F11K02, F12K02, F13K02, F13K03.
  - The fields that belong to both the fully concatenated key and the concatenated key. For example, the following fields of entity segment S13 belong to its concatenated key: F11K01, F11K02, F12K02.
- For each target DB2 table:
  - The columns that belong to the DB2 primary key. For example, the following fields of table T13 belong to its DB2 primary key: C13K01, C13K02, C13K03, C13K05, C13K06.
  - The columns that belong to both the DB2 foreign key and the DB2 primary key. For example, the following columns of table T13 belong to its DB2 foreign key: C13K01, C13K02, C13K03.

The figure also shows that the fully concatenated key of each entity segment is mapped to the DB2 primary key of the target table. For example, the fields F11K01, F11K02, F12K02, F13K02, and F13K03 belong to the fully concatenated key of

entity segment S13 and are mapped to the following columns of the DB2 primary key: C13K01, C13K02, C13K03, C13K05, C13K06.

Additionally, the concatenated key of each entity segment is mapped to the DB2 foreign key of the target table. For example, the fields F11K01, F11K02, and F12K02 belong to the concatenated key of entity segment S13 and are mapped to the following columns of the DB2 foreign key: C13K01, C13K02, C13K03.

Note: In our example, S16 is assumed to have multiple occurrences under its parent and no IMS key field. Therefore you cannot map its fully concatenated key to a unique DB2 primary key. However, S16 is assumed to have an identification field (F16D03), which identifies S16 uniquely under its parent.

The combination of the fields from S16's fully concatenated key and F16D03 is mapped to the columns of the DB2 primary key: F11K01, F11K02, F12K02, and F16D03 are mapped to columns C16K01, C16K02, C16K03, and C16K06.

Field Mapping Overview: 5

Table 52 through Table 56 on page 709 show the characteristics of the source IMS fields and the target DB2 columns. The tables contain the following information:

- For the IMS source field:
  - The name of the IMS field (as defined to IMS DPROP either on the NAME keyword of the DataRefresher FIELD statement, or in the FLDNAME column of the DPRIFLD MVG input table)
  - The data type (as defined to IMS DPROP either on the TYPE keyword of the DataRefresher FIELD statement, or in the DATATYPE column of the DPRIFLD MVG input table)
  - The length in bytes (as defined to IMS DPROP either on the BYTES keyword of the DataRefresher FIELD statement or in the BYTES column of the DPRIFLD MVG input table)
- For the DB2 target column:
  - The name of the target DB2 column (as defined to DB2 in the CREATE TABLE statement)
  - The data type of the target DB2 column (as defined to DB2 in the CREATE TABLE statement)

For a decimal column, specify the number of digits of the column in the CREATE TABLE statement. However, for a packed IMS field, do not specify the number of digits; instead, specify the number of bytes.

Table 52. Source Segment: S11 - Target Table: T11

Source IMS field			**Target DB2 column**	
Field Name	**Data Type**	**Length (in bytes)**	**Column Name**	**Data Type**
F11K01	C	2	C11K01	CHAR(2)
F11K02	P	4	C11K02	DECIMAL(7)
F11D03	C	4	C11D03	CHAR(4)
F11D04	C	8	C11D04	CHAR(8)
F11D05	P	6	C11D05	DECIMAL(11)

Table 53. Source Segment: S12 (and Its Concatenated Key) - Target Table: T12

Source IMS field			Target DB2 column	
Field Name	**Data Type**	**Length (in bytes)**	**Column Name**	**Data Type**
F11K01	C	2	C12K01	CHAR(2)
F11K02	P	4	C12K02	DECIMAL(7)
F12D01	C	1	C12D03	CHAR(1)
F12K02	P	4	C12K04	DECIMAL(7)
F12D03	C	20	C12D05	CHAR(20)
F12D04	C	1	C12D06	CHAR(1)

Table 54. Source Segments: S13 (and Its Concatenated Key), S14 and S15 - Target Table: T13

Source IMS field			Target DB2 column	
Field Name	**Data Type**	**Length (in bytes)**	**Column Name**	**Data Type**
F11K01	C	2	C13K01	CHAR(2)
F11K02	P	4	C13K02	DECIMAL(7)
F12K02	P	4	C13K03	DECIMAL(7)
F13D01	H	2	C13D04	SMALLINT
F13K02	P	3	C13K05	DECIMAL(5)
F13K03	P	5	C13K06	DECIMAL(9)
F13D04	H	2	C13D07	SMALLINT
F13D05	F	4	C13D08	INTEGER
F13D06	H	2	C13D09	SMALLINT
F14D01	C	2	C13D10	CHAR(2)
F14D02	C	5	C13D11	CHAR(5)
F14D03	C	8	C13D12	CHAR(8)
F15D01	P	2	C13D13	DECIMAL(3)
F15D02	C	30	C13D14	CHAR(30)

Table 55. Source Segment: S16 (and Its Concatenated Key) - Target Table: T16

Source IMS field			Target DB2 column	
Field Name	**Data Type**	**Length (in bytes)**	**Column Name**	**Data Type**
F11K01	C	2	C16K01	CHAR(2)
F11K02	P	4	C16K02	DECIMAL(7)
F12K02	P	4	C16K03	DECIMAL(7)
F16D01	F	4	C16D04	INTEGER
F16D02	F	4	C16D05	INTEGER
F16D03	H	2	C16K06	SMALLINT

Table 56. Source Segments: S17 (and Its Concatenated Key) and S18 - Target Table: T17

Source IMS field			Target DB2 column	
Field Name	**Data Type**	**Length (in bytes)**	**Column Name**	**Data Type**
F11K01	C	2	C17K01	CHAR(2)
F11K02	P	4	C17K02	DECIMAL(7)
F12K02	P	4	C17K03	DECIMAL(7)
F17K01	C	20	C17K04	CHAR(20)
F17D02	P	3	C17D05	DECIMAL(5)
F17D03	P	1	C17D06	DECIMAL(1)
F17D04	H	2	-[1]	-
F17D05	VC[2]	28	C17D07	VARCHAR(28)
F18D01	C	50	C17D08	CHAR(50)

1 F17D04 is the length field of variable-length field F17D05 and therefore has no target DB2 column.

2 F17D05 is a variable-length character field. Its length field is F17D04. If you are using DataRefresher to define the propagation request, you must indicate on the LENFIELD keyword that F17D014 is the length field of F17D05. If you are defining the propagation request in the MVG input tables, you must specify that F17D04 is the length field of F17D05 in the LENFIELD column of the DPRIFLD table.

DBD Definitions: 5

The DBD control statements for the model database are described in Figure 572 on page 710.

```
DBD     NAME=D01,ACCESS=HISAM, EXIT=(EKYRUP00)
DATASET DD1=DDNAME1,OVFLW=DDNAME2,DEVICE=3380

SEGM    NAME=S11,PARENT=0,BYTES=24
FIELD   NAME=(F11KEY,SEQ,U),TYPE=C,BYTES=6,START=1
FIELD   NAME=F11K01,TYPE=C,BYTES=2,START=1
FIELD   NAME=F11K02,TYPE=P,BYTES=4,START=3
FIELD   NAME=F11D03,TYPE=C,BYTES=4,START=7
FIELD   NAME=F11D04,TYPE=C,BYTES=8,START=11
FIELD   NAME=F11D05,TYPE=P,BYTES=6,START=19

SEGM    NAME=S12,PARENT=S11,BYTES=26
FIELD   NAME=(F12KEY,SEQ,U),TYPE=C,BYTES=4,START=2
FIELD   NAME=F12D01,TYPE=C,BYTES=1,START=1
FIELD   NAME=F12K02,TYPE=P,BYTES=4,START=2
FIELD   NAME=F12D03,TYPE=C,BYTES=20,START=6
FIELD   NAME=F12D04,TYPE=C,BYTES=1,START=26

SEGM    NAME=S13,PARENT=S12,BYTES=18
FIELD   NAME=(F13KEY,SEQ,U),TYPE=C,BYTES=8,START=3
FIELD   NAME=F13D01,TYPE=X,BYTES=2,START=1
FIELD   NAME=F13K02,TYPE=P,BYTES=3,START=3
FIELD   NAME=F13K03,TYPE=P,BYTES=5,START=6
FIELD   NAME=F13D04,TYPE=X,BYTES=2,START=11
FIELD   NAME=F13D05,TYPE=X,BYTES=4,START=13
FIELD   NAME=F13D06,TYPE=X,BYTES=2,START=17

SEGM    NAME=S14,PARENT=S13,BYTES=15
FIELD   NAME=F14D01,TYPE=C,BYTES=2,START=1
FIELD   NAME=F14D02,TYPE=C,BYTES=5,START=3
FIELD   NAME=F14D03,TYPE=X,BYTES=8,START=8

SEGM    NAME=S15,PARENT=S13,BYTES=32
FIELD   NAME=F15D01,TYPE=P,BYTES=2,START=1
FIELD   NAME=F15D02,TYPE=C,BYTES=30,START=3

SEGM    NAME=S16,PARENT=S12,BYTES=10
FIELD   NAME=F16D01,TYPE=X,BYTES=4,START=1
FIELD   NAME=F16D02,TYPE=X,BYTES=4,START=5
FIELD   NAME=F16D03,TYPE=X,BYTES=2,START=9

SEGM    NAME=S17,PARENT=S12,BYTES=54
FIELD   NAME=(F17KEY,SEQ,U),TYPE=C,BYTES=20,START=1
FIELD   NAME=F17K01,TYPE=C,BYTES=20,START=1
FIELD   NAME=F17D02,TYPE=P,BYTES=3,START=21
FIELD   NAME=F17D03,TYPE=P,BYTES=1,START=24
FIELD   NAME=F17D04,TYPE=X,BYTES=2,START=25
FIELD   NAME=F17D05,TYPE=C,BYTES=28,START=27

SEGM    NAME=S18,PARENT=S17,BYTES=50
FIELD   NAME=F18D01,TYPE=C,BYTES=50,START=1

DBDGEN
FINISH
END
```

Figure 572. DBD control statements for the Model Database (Example 5)

The EXIT keyword in the DBD control statement tells IMS to give control to EKYRUP00, the IMS DPROP-provided Data Capture Exit routine, when a DL/I call updates a segment of this DBD.

Usually, IMS DPROP and DataRefresher do not require you to define the non-key fields in the IMS DBD. They will not use such definitions.

Table Definitions: 5

The SQL statements to define the target tables are shown in Figure 573 through Figure 577 on page 713. These examples illustrate RIRs between the DB2 tables. The defined RIRs match the IMS parent/child relationships between the IMS source segments:

- The DB2 primary key of a target table is the propagated fully concatenated key of the IMS source segment.
- The DB2 foreign key of a target table is the propagated concatenated key of the IMS source segment

```
CREATE TABLE prefix.T11
          (C11K01   CHAR(2)       NOT NULL,
           C11K02   DECIMAL(7)    NOT NULL,
           C11D03   CHAR(4)       NOT NULL WITH DEFAULT,
           C11D04   CHAR(8)       NOT NULL WITH DEFAULT,
           C11D05   DECIMAL(11)   NOT NULL WITH DEFAULT,
       PRIMARY KEY (C11K01,C11K02))
       IN database.tablespace;

CREATE UNIQUE INDEX prefix.TX11
       ON prefix.T11 (C11K01,C11K02);
```

Figure 573. Table T11

```
CREATE TABLE prefix.T12
          (C12K01   CHAR(2)       NOT NULL,
           C12K02   DECIMAL(7)    NOT NULL,
           C12D03   CHAR(1)       NOT NULL WITH DEFAULT,
           C12K04   DECIMAL(7)    NOT NULL,
           C12D05   CHAR(20)      NOT NULL WITH DEFAULT,
           C12D06   CHAR(1)       NOT NULL WITH DEFAULT,
       PRIMARY KEY (C12K01,C12K02,C12K04),
       FOREIGN KEY (C12K01,C12K02)
               REFERENCES prefix.T11 ON DELETE CASCADE)
       IN database.tablespace;

CREATE UNIQUE INDEX prefix.TX12
       ON prefix.T12 (C12K01,C12K02,C12K04);
```

Figure 574. Table T12

```
CREATE TABLE prefix.T13
          (C13K01   CHAR(2)       NOT NULL,
           C13K02   DECIMAL(7)    NOT NULL,
           C13K03   DECIMAL(7)    NOT NULL,
           C13D04   SMALLINT      NOT NULL WITH DEFAULT,
           C13K05   DECIMAL(5)    NOT NULL,
           C13K06   DECIMAL(9)    NOT NULL,
           C13D07   SMALLINT      NOT NULL WITH DEFAULT,
           C13D08   INTEGER       NOT NULL WITH DEFAULT,
           C13D09   SMALLINT      NOT NULL WITH DEFAULT,
           C13D10   CHAR(2)       NOT NULL WITH DEFAULT,
           C13D11   CHAR(5)       NOT NULL WITH DEFAULT,
           C13D12   CHAR(8)       NOT NULL WITH DEFAULT,
           C13D13   DECIMAL(3)    NOT NULL WITH DEFAULT,
           C13D14   CHAR(30)      NOT NULL WITH DEFAULT,
       PRIMARY KEY (C13K01,C13K02,C13K03,C13K05,C13K06),
       FOREIGN KEY (C13K01,C13K02,C13K03)
               REFERENCES prefix.T12 ON DELETE CASCADE)
       IN database.tablespace;

CREATE UNIQUE INDEX prefix.TX13
       ON prefix.T13 (C13K01,C13K02,C13K03,C13K05,C13K06);
```

Figure 575. Table T13

```
CREATE TABLE prefix.T16
          (C16K01   CHAR(2)       NOT NULL,
           C16K02   DECIMAL(7)    NOT NULL,
           C16K03   DECIMAL(7)    NOT NULL,
           C16D04   INTEGER       NOT NULL WITH DEFAULT,
           C16D05   INTEGER       NOT NULL WITH DEFAULT,
           C16K06   SMALLINT      NOT NULL,
       PRIMARY KEY (C16K01,C16K02,C16K03,C16K06),
       FOREIGN KEY (C16K01,C16K02,C16K03)
               REFERENCES prefix.T102 ON DELETE CASCADE)
       IN database.tablespace;

CREATE UNIQUE INDEX prefix.TX16
       ON prefix.T16 (C16K01, C16K02, C16K03, C16K06);
```

Figure 576. Table T16

```
CREATE TABLE prefix.T17
          (C17K01   CHAR(2)       NOT NULL,
           C17K02   DECIMAL(7)    NOT NULL,
           C17K03   DECIMAL(7)    NOT NULL,
           C17K04   CHAR(20)      NOT NULL,
           C17D05   DECIMAL(5)    NOT NULL WITH DEFAULT,
           C17D06   DECIMAL(1)    NOT NULL WITH DEFAULT,
           C17D07   VARCHAR(28)   NOT NULL WITH DEFAULT,
           C17D08   CHAR(50)      NOT NULL WITH DEFAULT,
       PRIMARY KEY (C17K01,C17K02,C17K03,C17K04),
       FOREIGN KEY (C17K01,C17K02,C17K03)
               REFERENCES prefix.T12 ON DELETE CASCADE)
       IN database.tablespace;

CREATE UNIQUE INDEX prefix.TX17
       ON prefix.T17 (C17K01, C17K02, C17K03, C17K04);
```

Figure 577. Table T17

Defining Propagation Requests with DataRefresher: 5

Defining the DXTPSB: 5

DataRefresher requires that you describe the IMS database to DataRefresher by providing a CREATE DXTPSB command. The DataRefresher control statements for our model database are shown in Figure 578 on page 714.

```
                                                                          NOTE
CREATE  DXTPSB   NAME=PSB1
        DXTPCB   NAME=PSB1PCB1,DBACCESS=HISAM,DBNAME=D01

       SEGMENT   NAME=S11,PARENT=0,BYTES=24
         FIELD   NAME=F11KEY,TYPE=C,BYTES=6,START=1,SEQFLD=R
         FIELD   NAME=F11K01,TYPE=C,BYTES=2,START=1
         FIELD   NAME=F11K02,TYPE=P,BYTES=4,START=3
         FIELD   NAME=F11D03,TYPE=C,BYTES=4,START=7
         FIELD   NAME=F11D04,TYPE=C,BYTES=8,START=11
         FIELD   NAME=F11D05,TYPE=P,BYTES=6,START=19

       SEGMENT   NAME=S12,PARENT=S11,BYTES=26
         FIELD   NAME=F12D01,TYPE=C,BYTES=1,START=1
         FIELD   NAME=F12K02,TYPE=P,BYTES=4,START=2
         FIELD   NAME=F12D03,TYPE=C,BYTES=20,START=6
         FIELD   NAME=F12D04,TYPE=C,BYTES=1,START=26

       SEGMENT   NAME=S13,PARENT=S12,BYTES=18
         FIELD   NAME=F13D01,TYPE=H,BYTES=2,START=1
         FIELD   NAME=F13K02,TYPE=P,BYTES=3,START=3
         FIELD   NAME=F13K03,TYPE=P,BYTES=5,START=6
         FIELD   NAME=F13D04,TYPE=H,BYTES=2,START=11
         FIELD   NAME=F13D05,TYPE=F,BYTES=4,START=13
         FIELD   NAME=F13D06,TYPE=H,BYTES=2,START=17

       SEGMENT   NAME=S14,PARENT=S13,BYTES=15,MAXNBR=1
         FIELD   NAME=F14D01,TYPE=C,BYTES=2,START=1
         FIELD   NAME=F14D02,TYPE=C,BYTES=5,START=3
         FIELD   NAME=F14D03,TYPE=C,BYTES=8,START=8

       SEGMENT   NAME=S15,PARENT=S13,BYTES=32,MAXNBR=1
         FIELD   NAME=F15D01,TYPE=P,BYTES=2,START=1
         FIELD   NAME=F15D02,TYPE=C,BYTES=30,START=3

       SEGMENT   NAME=S16,PARENT=S12,BYTES=10
         FIELD   NAME=F16D01,TYPE=F,BYTES=4,START=1
         FIELD   NAME=F16D02,TYPE=F,BYTES=4,START=5
         FIELD   NAME=F16D03,TYPE=H,BYTES=2,START=9

       SEGMENT   NAME=S17,PARENT=S12,BYTES=54
         FIELD   NAME=F17K01,TYPE=C,BYTES=20,START=1
         FIELD   NAME=F17D02,TYPE=P,BYTES=3,START=21
         FIELD   NAME=F17D03,TYPE=P,BYTES=1,START=24
         FIELD   NAME=F17D04,TYPE=H,BYTES=2,START=25                        1
         FIELD   NAME=F17D05,TYPE=VC,BYTES=28,START=27,LFIELD=F17D04        2

       SEGMENT   NAME=S18,PARENT=S17,BYTES=50,MAXNBR=1
         FIELD   NAME=F18D01,TYPE=C,BYTES=50,START=1;
```

Figure 578. DataRefresher control statements for the Model Database (Example 5)

Notes:

1. F17D04 is the length field of variable-length field F17D05. It is not mapped by the EXTRACT statement to a DB2 column.
2. F17D05 is a variable-length character field. Its length field is F17D04. You must indicate on the LENFIELD keyword that F17D014 is the length field of F17D05.

Defining the DXTVIEWs: 5

DataRefresher requires a view description for each hierarchical path of the database. The DataRefresher control statements used to define the views for our

example database are shown in Figure 579 through Figure 582.

```
CREATE  DXTVIEW NAME=VIEWA,
                DXTPSB=PSB1,DXTPCB=PSB1PCB1,SEGMENT=S14,MINSEGM=S13,
                FIELDS=(F11K01,F11K02,F11D03,F11D04,F11D05,
                        F12D01,F12K02,F12D03,F12D04,
                        F13D01,F13K02,F13K03,F13D04,F13D05,F13D06,
                        F14D01,F14D02,F14D03);
```

Figure 579. View A for Segments S11, S12, S13, and S14

```
CREATE  DXTVIEW NAME=VIEWB,
                DXTPSB=PSB1,DXTPCB=PSB1PCB1,SEGMENT=S15,MINSEGM=S13,
                FIELDS=(F11K01,F11K02,F11D03,F11D04,F11D05,
                        F12D01,F12K02,F12D03,F12D04,
                        F13D01,F13K02,F13K03,F13D04,F13D05,F13D06,
                        F15D01,F15D02);
```

Figure 580. View B for Segments S11, S12, S13, and S15

```
CREATE  DXTVIEW NAME=VIEWC,
                DXTPSB=PSB1,DXTPCB=PSB1PCB1,SEGMENT=S16,MINSEGM=S16,
                FIELDS=(F11K01,F11K02,F11D03,F11D04,F11D05,
                        F12D01,F12K02,F12D03,F12D04,
                        F16D01,F16D02,F16D03);
```

Figure 581. View C for Segments S11, S12, and S16

```
CREATE  DXTVIEW NAME=VIEWD,
                DXTPSB=PSB1,DXTPCB=PSB1PCB1,SEGMENT=S18,MINSEGM=S17,
                FIELDS=(F11K01,F11K02,F11D03,F11D04,F11D05,
                        F12D01,F12K02,F12D03,F12D04,
                        F17K01,F17D02,F17D03,F17D04,F17D05,
                        F18D01);
```

Figure 582. View D for Segments S11, S12, S17, and S18

Providing DataRefresher UIM SUBMIT Commands and EXTRACT Statements: 5

Figure 583 on page 716 through Figure 587 on page 720 illustrate the DataRefresher SUBMIT commands and EXTRACT statements used to define the propagation requests.

The DataRefresher SUBMIT/EXTRACT creates the propagation requests in the IMS DPROP directory but does not load the data to the DB2 tables. This is the responsibility of the DataRefresher DEM. Refer to “Providing DataRefresher DEM control statements: 5 ” on page 720 for examples of the DataRefresher DEM control statement.

```
SUBMIT   EXTID=PR1,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               USERDECK='ENFORCE NO',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,
                         MAPCASE=1,
                         MAPDIR=HR,
                         ERROPT=BACKOUT,
                         ACTION=REPL'

EXTRACT  INTO   prefix.T11 (C11K01 NOT NULL,
                            C11K02 NOT NULL,
                            C11D03 NOT NULL WITH DEFAULT,
                            C11D04 NOT NULL WITH DEFAULT,
                            C11D05 NOT NULL WITH DEFAULT)

         OPTIONS            (FLDERR(HALT))

         SELECT             VIEWA.F11K01,
                            VIEWA.F11K02,
                            VIEWA.F11D03,
                            VIEWA.F11D04,
                            VIEWA.F11D05

         FROM               VIEWA;
```

Figure 583. DataRefresher SUBMIT/EXTRACT for Table T11

```
SUBMIT   EXTID=PR2,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               USERDECK='ENFORCE NO',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,
                         MAPCASE=1,
                         MAPDIR=HR,
                         ERROPT=BACKOUT,
                         ACTION=REPL'

EXTRACT  INTO   prefix.T12 (C12K01 NOT NULL,
                            C12K02 NOT NULL,
                            C12D03 NOT NULL WITH DEFAULT,
                            C12K04 NOT NULL,
                            C12D05 NOT NULL WITH DEFAULT,
                            C12D06 NOT NULL WITH DEFAULT)

         OPTIONS            (FLDERR(HALT))

         SELECT             VIEWA.F11K01,
                            VIEWA.F11K02,
                            VIEWA.F12D01,
                            VIEWA.F12K02,
                            VIEWA.F12D03,
                            VIEWA.F12D04

         FROM               VIEWA;
```

Figure 584. DataRefresher SUBMIT/EXTRACT for Table T12

```
SUBMIT   EXTID=PR3,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               USERDECK='ENFORCE NO',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,
                         MAPCASE=2,
                         MAPDIR=HR,
                         ERROPT=BACKOUT,
                         ACTION=REPL'

EXTRACT  INTO   prefix.T13 (C13K01 NOT NULL,
                            C13K02 NOT NULL,
                            C13K03 NOT NULL,
                            C13D04 NOT NULL WITH DEFAULT,
                            C13K05 NOT NULL,
                            C13K06 NOT NULL,
                            C13D07 NOT NULL WITH DEFAULT,
                            C13D08 NOT NULL WITH DEFAULT,
                            C13D09 NOT NULL WITH DEFAULT,
                            C13D10 NOT NULL WITH DEFAULT,
                            C13D11 NOT NULL WITH DEFAULT,
                            C13D12 NOT NULL WITH DEFAULT,
                            C13D13 NOT NULL WITH DEFAULT,
                            C13D14 NOT NULL WITH DEFAULT)

         OPTIONS            (FLDERR(HALT))

         SELECT             VIEWA.F11K01,
                            VIEWA.F11K02,
                            VIEWA.F12K02,
                            VIEWA.F13D01,
                            VIEWA.F13K02,
                            VIEWA.F13K03,
                            VIEWA.F13D04,
                            VIEWA.F13D05,
                            VIEWA.F13D06,
                            VIEWA.F14D01,
                            VIEWA.F14D02,
                            VIEWA.F14D03,
                            VIEWB.F15D01,
                            VIEWB.F15D02

         FROM               VIEWA,
                            VIEWB;
```

Figure 585. DataRefresher SUBMIT/EXTRACT for Table T13

```
SUBMIT   EXTID=PR6,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               USERDECK='ENFORCE NO',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,
                         MAPCASE=1,
                         MAPDIR=HR,
                         ERROPT=BACKOUT,
                         ACTION=REPL'

EXTRACT  INTO   prefix.T16 (C16K01 NOT NULL,
                            C16K02 NOT NULL,
                            C16K03 NOT NULL,
                            C16D04 NOT NULL WITH DEFAULT,
                            C16D05 NOT NULL WITH DEFAULT,
                            C16K06 NOT NULL)

         OPTIONS            (FLDERR(HALT))

         SELECT             VIEWC.F11K01,
                            VIEWC.F11K02,
                            VIEWC.F12K02,
                            VIEWC.F16D01,
                            VIEWC.F16D02,
                            VIEWC.F16D03

         FROM               VIEWC;
```

Figure 586. DataRefresher SUBMIT/EXTRACT for Table T16

```
SUBMIT   EXTID=PR7,
               CD=JCS,
               JCS=DDJCS01,
               FORMAT=SOURCE,
               USERDECK=ENFORCE NO',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,
                         MAPCASE=2,
                         MAPDIR=HR,
                         ERROPT=BACKOUT,
                         ACTION=REPL'

EXTRACT  INTO   prefix.T17 (C17K01 NOT NULL,
                            C17K02 NOT NULL,
                            C17K03 NOT NULL,
                            C17K04 NOT NULL,
                            C17D05 NOT NULL WITH DEFAULT,
                            C17D06 NOT NULL WITH DEFAULT,
                            C17D07 NOT NULL WITH DEFAULT,
                            C17D08 NOT NULL WITH DEFAULT)

         OPTIONS            (FLDERR(HALT))

         SELECT             VIEWD.F11K01,
                            VIEWD.F11K02,
                            VIEWD.F12K02,
                            VIEWD.F17K01,
                            VIEWD.F17D02,
                            VIEWD.F17D03,
                            VIEWD.F17D05,
                            VIEWD.F18D01

         FROM               VIEWD;
```

Figure 587. DataRefresher SUBMIT/EXTRACT for Table T17

Providing DataRefresher DEM control statements: 5

Figure 588 shows an example of a DataRefresher DEM control statement that you can use to extract the data from the IMS database.

```
INITDEM NAME=DEM1
USE DXTPSB=PSB1;
```

Figure 588. DataRefresher DEM control statements (Example 5)

Defining Propagation Requests with the MVG Input Tables ISPF Application: 5

In this example, five different propagation requests are defined on the MVG input tables. The entire panel flow is shown for each propagation request.

Call the MVGIN Front End Application as described in Chapter 20, “CCU Front End ,” on page 523. The panel shown in Figure 589 on page 721 is displayed:

```
EKYMP00E ------------- DPROP MVG Input Tables Application ---------------------
Command ===>

                                                              DPROP Environ-
   1 - Insert, Update, Show, Delete a Propagation Request     ment  : (name)
   2 - Propagation Request Overview to Update, Show,          DPROP-
       Delete a Propagation Request                           System: (name)
   3 - Setup IMS and DPROP System Environment                 DPROP-System-
   4 - Edit existing MVG Input Tables Insert/Delete Jobs      Number: (number)

   E - Return


     Enter the fully qualified name of the partitioned data set to be
     used to store the generated MVGIN job control:
        Data Set Name ===>
     Enter the volume serial name to be used to allocate a new data set:
        Volume Serial ===>


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 589. EKYMP00E: DPROP MVG Input Tables Application (Example 5)

The first time you call the MVGIN Front End Application, you have to complete the setup panels and enter a data set name in the Data Set Name field.

Panel Flow for PR1: 5

After completing the setup, select option 1 to insert a new propagation request. The panel shown in Figure 590 is displayed:

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                     DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR1


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                 ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Figure 590. EKYMP10E: Maintain a Propagation Request (Example 5 - PR1)

Type I in the Option field and PR1 in the propagation request Name field and press Enter.

After your input is validated, the panel shown in Figure 591 on page 722 is displayed.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR1

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 591. EKYMP15E: Propagation Request Definition Selection (Example 5 - PR1)

Select option 1 to enter the general propagation request specifications. The panel shown in Figure 592 is displayed.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------
Command ===>
                                                Propagation Request: PR1

                                                                    More:   - +
  PR Type                   ===> E         ( E, L, F or U                   )
  Mapping Case              ===> 1         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===>           ( ID, DENORM or blank            )
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <
  User ID                   ===>           ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 592. EKYMP20E: Propagation Request (Example 5 - PR1)

Type END on the command line and press Enter to return to panel EKYMP15E (shown in Figure 591 on page 722) with the state of option 1 changed to `processed`.

Select option 2 on panel EKYMP15E to define the DB2 table. The panel shown in Figure 593 is displayed.

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                  Propagation Request: PR1

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===>            ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> T11                 ( a name up to 18 bytes )



         Note: The DB2 table you enter is not checked for existence.





Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 593. EKYMP30E: DB2 Table (Example 5 - PR1)

Type END on the command line and press Enter to return to panel EKYMP15E (shown in Figure 591 on page 722) with the state of options 1 and 2 changed to `processed`.

Select option 3 on panel EKYMP15E to define the segments, fields and columns. The panel shown in Figure 594 is displayed.

```
EKYMP40E --- DPROP MVGIN - Entity Segment for a Mapping Case 1 PR -------------
Command ===>

    F - Fields for the entity segment            Propagation Request: PR1
    P - Fields for the parent segments           Database-
                                                 Organization: HISAM/VSAM

  IMS Database Name         ===> D01       ( a name up to 8 bytes           )
  Segment Name              ===> S11       ( a name up to 8 bytes           )


  PCB Label Name            ===>           ( a name up to 8 bytes           )


  Segment Exit Routine      ===>           ( a name up to 8 bytes, optional )
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760             )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                         )


Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 594. EKYMP40E: Entity Segment for a Mapping Case 1 PR Propagation Request (Example 5 - PR1)

Because segment S11 is the root segment, you cannot define fields for parent segments.

To define the fields for the entity segment, type F on the command line and press Enter. The panel shown in Figure 595 is displayed.

```
EKYMP41E -------- DPROP MVGIN - Entity Segment Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                   Propagation Request: PR1
    S - Sort in POSITION sequence               Entity Segment:      S11

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
FLDNAME  ===> F11KEY                            POSITION ===> 1
BYTES    ===> 6         DATATYPE ===> C         DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    2 ------------------------------------------------------------
FLDNAME  ===> F11K01                            POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> C         DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C11K01
-------------    3 ------------------------------------------------------------
FLDNAME  ===> F11K02                            POSITION ===> 3
BYTES    ===> 4         DATATYPE ===> P         DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C11K02
-------------    4 ------------------------------------------------------------
FLDNAME  ===> F11D03                            POSITION ===> 7
BYTES    ===> 4         DATATYPE ===> C         DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C11D03
-------------    5 ------------------------------------------------------------
FLDNAME  ===> F11D04                            POSITION ===> 11
BYTES    ===> 8         DATATYPE ===> C         DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C11D04
-------------    6 ------------------------------------------------------------
FLDNAME  ===> F11D05                            POSITION ===> 19
BYTES    ===> 6         DATATYPE ===> P         DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C11D05
```

Figure 595. EKYMP41E: Entity Segment Fields/Columns (Example 5 - PR1)

Type END on the command line and press Enter to return to panel EKYMP40E (shown in Figure 594 on page 723) Repeat the procedure to return to panel EKYMP15E (shown in Figure 591 on page 722) with the state of options 1, 2, and 3 changed to `processed`.

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. The panel shown in Figure 596 is overlaid on panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

        _________DPROP MVGIN - Generate MVG Input Tables Insert Job____    ame)
       | EKYMPIIE ==>                                                  |  1
       |                                                               |
       |                                DPROP-System:        (name)    |  s
    1  |                                Propagation Request: PR1       |  d *
    2  |                                                               |  d *
    3  |  Enter the member name to be used to store the generated      |  d *
       |  MVG input tables insert job.  The job is stored in data set  |  d *
    4  |  "(library)".                                                 |  d *
       |                                                               |
    5  |     Member Name    ===> PR1                                   |
       |                                                               |
    S  | Press <ENTER> to continue or enter END command to return.     |
       |                                                               |
    E  |                                                               |
       |_______________________________________________________________|


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 596. Generate MVG Input Tables Insert Job (Example 5 - PR1)

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Return to panel EKYMP10E to define PR2.

Panel Flow for PR2: 5

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                        DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR2


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                 ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Figure 597. EKYMP10E: Maintain a Propagation Request (Example 5 - PR2)

Type I in the Option field and PR2 in the propagation request Name field and press Enter.

After your input is validated, the panel shown in Figure 598 is displayed.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR2

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 598. EKYMP15E: Propagation Request Definition Selection (Example 5 - PR2)

Select option 1 to enter the general propagation request specifications. The panel shown in Figure 599 on page 727 is displayed.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------
Command ===>
                                                Propagation Request: PR2

                                                                    More:   - +
  PR Type                   ===> E         ( E, L, F or U                   )
  Mapping Case              ===> 1         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===>           ( ID, DENORM or blank            )
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                       <
    >                                                                       <
    >                                                                       <
    >                                                <
  User ID                   ===>           ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                       <
    >                                                                       <
    >                                                                       <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 599. EKYMP20E: Propagation Request (Example 5 - PR2)

Type END on the command line and press Enter to return to panel EKYMP15E (shown in Figure 591 on page 722) with the state of options 1 changed to `processed`.

Select option 2 on panel EKYMP15E to define the DB2 table. The panel shown in Figure 600 on page 728 is displayed.

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                   Propagation Request: PR2

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===>            ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> T12                  ( a name up to 18 bytes )


         Note: The DB2 table you enter is not checked for existence.





Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 600. EKYMP30E: DB2 Table (Example 5 - PR2)

Type END on the command line and press Enter to return to panel EKYMP15E (shown in Figure 591 on page 722) with the state of options 1 and 2 changed to processed.

Select option 3 on panel EKYMP15E to define the segments, fields and columns. The panel shown in Figure 601 is displayed.

```
EKYMP40E --- DPROP MVGIN - Entity Segment for a Mapping Case 1 PR -------------
Command ===>

    F - Fields for the entity segment            Propagation Request: PR2
    P - Fields for the parent segments           Database-
                                                 Organization: HISAM/VSAM

  IMS Database Name         ===> D01       ( a name up to 8 bytes           )
  Segment Name              ===> S12       ( a name up to 8 bytes           )


  PCB Label Name            ===>           ( a name up to 8 bytes           )


  Segment Exit Routine      ===>           ( a name up to 8 bytes, optional )
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760             )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                         )


Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 601. EKYMP40E: Entity Segment for a Mapping Case 1 PR (Example 5 - PR2)

To define the fields for the parent segments, enter P on the command line and press Enter. The panel shown in Figure 602 on page 729 is displayed.

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR2
        sequence                             Entity/Internal Segment: S12

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11KEY                              POSITION ===> 1
BYTES    ===> 6         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11K01                              POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C12K01
-------------    3 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11K02                              POSITION ===> 3
BYTES    ===> 4         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C12K02
-------------    4 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D03                              POSITION ===> 7
BYTES    ===> 4         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    5 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D04                              POSITION ===> 11
BYTES    ===> 8         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    6 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D05                              POSITION ===> 19
BYTES    ===> 6         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
```

Figure 602. EKYMPPFE: Parent Segments Fields/Columns (Example 5 - PR2)

Type END on the command line and press Enter to return to panel EKYMP40E (shown in Figure 601 on page 728).

To define the fields for the entity segment, enter F on the command line of panel EKYMP40E and press Enter. The panel shown in Figure 603 on page 730 is displayed.

```
EKYMP41E -------- DPROP MVGIN - Entity Segment Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR2
    S - Sort in POSITION sequence                Entity Segment:      S12

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
FLDNAME  ===> F12D01                             POSITION ===> 1
BYTES    ===> 1         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C12D03
-------------    2 ------------------------------------------------------------
FLDNAME  ===> F12K02                             POSITION ===> 2
BYTES    ===> 4         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C12K04
-------------    3 ------------------------------------------------------------
FLDNAME  ===> F12D03                             POSITION ===> 6
BYTES    ===> 20        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C12D05
-------------    4 ------------------------------------------------------------
FLDNAME  ===> F12D04                             POSITION ===> 26
BYTES    ===> 1         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C12D06
```

Figure 603. EKYMP41E: Entity Segment Fields/Columns (Example 5 - PR2)

Type END on the command line and press Enter to return to panel EKYMP40E (shown in Figure 601 on page 728).

Repeat the procedure to return to panel EKYMP15E. (shown in Figure 598 on page 726) with the state of options 1, 2, and 3 changed to `processed`.

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. The panel shown in Figure 604 on page 731 is overlaid on panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

          _________DPROP MVGIN - Generate MVG Input Tables Insert Job____   ame)
         | EKYMPIIE ==>                                                   | 2
         |                                                                |
         |                                 DPROP-System:        (name)    | s
    1    |                                 Propagation Request: PR2       | d *
    2    |                                                                | d *
    3    |  Enter the member name to be used to store the generated       | d *
         |  MVG input tables insert job.  The job is stored in data set   | d *
    4    |  "(library)".                                                  | d *
         |                                                                |
    5    |     Member Name    ===> PR2                                    |
         |                                                                |
    S    | Press <ENTER> to continue or enter END command to return.      |
         |                                                                |
    E    |                                                                |
         |________________________________________________________________|


Enter option or move cursor before desired option.  Enter END command to return.
```

Figure 604. Generate MVG Input Tables Insert Job (Example 5 - PR2)

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Type END on the command line and press Enter to return to panel EKYMP10E to define PR3.

Panel Flow for PR3: 5

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                        DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR3


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                 ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Figure 605. EKYMP10E: Maintain a Propagation Request

Enter I in the Option field and PR3 in the propagation request Name field and press Enter.

After your input is validated, the panel shown in Figure 606 on page 732 is displayed.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR3

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 606. EKYMP15E: Propagation Request Definition Selection (Example 5 - PR3)

Select option 1 to enter the general propagation request specifications. The panel shown in Figure 607 is displayed.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------
Command ===>
                                                Propagation Request: PR3

                                                                    More:   - +
  PR Type                   ===> E         ( E, L, F or U                  )
  Mapping Case              ===> 2         ( 1, 2 or 3                     )
  Mapping Direction         ===> HR        ( HR, RH or TW                  )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank            )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank      )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                 )
  Propagation Exit Routine  ===>           ( only for PR type "U"          )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank        )
  Type of Path Data         ===>           ( ID, DENORM or blank           )
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                 )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                 )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                       <
    >                                                                       <
    >                                                                       <
    >                                                <
  User ID                   ===>           ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                       <
    >                                                                       <
    >                                                                       <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 607. EKYMP20E: Propagation Request (Example 5 - PR3)

Type END and press Enter to return to the panel shown in Figure 606 on page 732 with the state of option 1 changed to `processed`.

Select option 2 on panel EKYMP15E to define the DB2 table. The panel shown in Figure 608 is displayed.

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                   Propagation Request: PR3

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===>            ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> T13                 ( a name up to 18 bytes )


         Note: The DB2 table you enter is not checked for existence.





Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 608. EKYMP30E: DB2 Table (Example 5 - PR3)

Type END and press Enter to return to the panel shown in Figure 606 on page 732 with the state of options 1 and 2 changed to `processed`.

Select option 3 on panel EKYMP15E to define the segments, fields and columns. The panel shown in Figure 609 is displayed.

```
EKYMP50E - DPROP MVGIN - Segments and Fields for a Mapping Case 2 PR ----------
Command ===>


                                                   Propagation Request: PR3

                                          State
    1 - Entity Segment              not processed

    2 - Extension Segments and      not processed
        Fields/Columns              not processed


    E - Return





Enter option or move cursor before desired option.
Enter END command to return.
```

Figure 609. EKYMP50E: Segments and Fields for a Mapping Case 2 PR (Example 5 - PR3)

Select option 1 to define the entity segment. The panel shown in Figure 610 on page 734 is displayed.

```
EKYMP51E --- DPROP MVGIN - Entity Segment for a Mapping Case 2 PR -------------
Command ===>

                                                  Propagation Request: PR3
                                                  Database-
                                                  Organization: HISAM/VSAM

  IMS Database Name         ===> D01       ( a name up to 8 bytes           )
  Segment Name              ===> S13       ( a name up to 8 bytes           )


  PCB Label Name            ===>           ( a name up to 8 bytes           )


  Segment Exit Routine      ===>           ( a name up to 8 bytes, optional )
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760             )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                         )


Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 610. EKYMP51E: Entity Segment for a Mapping Case 2 Propagation Request (Example 5 - PR3)

Type END and press Enter to return to the panel shown in Figure 609 on page 733 with the state of option 1 changed to `processed`.

Select option 2 to define the extension segments and the fields and columns. The panel shown in Figure 611 is displayed.

```
EKYMP52E ------------- DPROP MVGIN - Extension Segments -----------------------
Command ===>                                                  SCROLL ===> HALF

    F - Fields for the entity and ext. segments  Propagation Request: PR3
    P - Fields for the parent segments           Entity Segment:      S13

  - Enter an X in the X column below to define an extension segment
  - Remove the X from the X column to unselect an extension segment

Press <ENTER> to validate your input.  Enter END command to return.

X  Seg Name  Seg Exit  Length of Segment processed  Format of Segment processed
                       by the Segment Exit Routine  by the Segment Exit Routine
                       ( from 1 up to 32760 )       ( F or V )

------------------------------------------------------------------------------  X  S14
------------------------------------------------------------------------------  X  S15
------------------------------------------------------------------------------
****************************** BOTTOM OF DATA ******************************
```

Figure 611. EKYMP52E: Extension Segments (Example 5 - PR3)

To define the fields for the parent segments, enter P on the command line and press ENTER. The panel shown in Figure 612 on page 735 is displayed.

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR3
        sequence                             Entity/Internal Segment: S13

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11KEY                             POSITION ===> 1
BYTES    ===> 6         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11K01                             POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13K01
-------------    3 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11K02                             POSITION ===> 3
BYTES    ===> 4         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13K02
-------------    4 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D03                             POSITION ===> 7
BYTES    ===> 4         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    5 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D04                             POSITION ===> 11
BYTES    ===> 8         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    6 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D05                             POSITION ===> 19
BYTES    ===> 6         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
```

Figure 612. EKYMPPFE: Parent Segments Fields/Columns (Example 5 - PR3) (Part 1 of 2)

Press F8 to display Figure 613 on page 736.

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR3
        sequence                             Entity/Internal Segment: S13

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    7 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12D01                              POSITION ===> 1
BYTES    ===> 1         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    8 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12K02                              POSITION ===> 2
BYTES    ===> 4         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13K03
-------------    9 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12D03                              POSITION ===> 6
BYTES    ===> 20        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------   10 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12D04                              POSITION ===> 26
BYTES    ===> 1         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
```

Figure 613. EKYMPPFE: Parent Segments Fields/Columns (Example 5 - PR3) (Part 2 of 2)

Type END and press Enter (or press F3) to return to the panel shown in Figure 611 on page 734.

To define the fields for the entity segment, enter F on the command line of panel EKYMP52E and press Enter. The panel shown in Figure 614 on page 737 is displayed.

```
EKYMP53E  DPROP MVGIN - Entity and Extension Segments Fields/Columns ----------
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR3
    S - Sort in SEGNAME, POSITION sequence       Entity Segment:      S13

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13D01                             POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D04
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13K02                             POSITION ===> 3
BYTES    ===> 3         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13K05
-------------    3 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13K03                             POSITION ===> 6
BYTES    ===> 5         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13K06
-------------    4 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13D04                             POSITION ===> 11
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D07
-------------    5 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13D05                             POSITION ===> 13
BYTES    ===> 4         DATATYPE ===> F          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D08
-------------    6 ------------------------------------------------------------
SEGNAME  ===> S13
FLDNAME  ===> F13D06                             POSITION ===> 17
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D09
```

Figure 614. EKYMP53E: Entity and Extension Segments Fields/Columns (Example 5 - PR3)

Press F8 to display the panel shown in Figure 615 on page 738.

```
EKYMP53E  DPROP MVGIN - Entity and Extension Segments Fields/Columns ----------
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR3
    S - Sort in SEGNAME, POSITION sequence       Entity Segment:      S13

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    7 ------------------------------------------------------------
SEGNAME  ===> S14
FLDNAME  ===> F14D01                             POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D10
-------------    8 ------------------------------------------------------------
SEGNAME  ===> S14
FLDNAME  ===> F14D02                             POSITION ===> 3
BYTES    ===> 5         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D11
-------------    9 ------------------------------------------------------------
SEGNAME  ===> S14
FLDNAME  ===> F14D03                             POSITION ===> 8
BYTES    ===> 8         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D12
-------------   10 ------------------------------------------------------------
SEGNAME  ===> S15
FLDNAME  ===> F15D01                             POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D13
-------------   11 ------------------------------------------------------------
SEGNAME  ===> S15
FLDNAME  ===> F15D02                             POSITION ===> 3
BYTES    ===> 30        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C13D14
```

Figure 615. EKYMP53E: Entity and Extension Segments Fields/Columns (Example 5 - PR3) (Part 2 of 2)

Enter END once to return to panel EKYMP52E.

Enter END again to return to panel EKYMP50E.

```
EKYMP50E - DPROP MVGIN - Segments and Fields for a Mapping Case 2 PR ----------
Command ===>


                                                Propagation Request: PR3

                                        State
    1 - Entity Segment           ***  processed  ***

    2 - Extension Segments and   ***  processed  ***
        Fields/Columns           ***  processed  ***


    E - Return





Enter option or move cursor before desired option.
Enter END command to return.
```

Enter END to return to panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR3

                                        State
    1 - Propagation Request      ***  processed  ***
    2 - DB2 Tables               ***  processed  ***
    3 - Segments and             ***  processed  ***
        Fields/Columns           ***  processed  ***
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. The following panel is displayed:

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

         _________DPROP MVGIN - Generate MVG Input Tables Insert Job____     ame)
        | EKYMPIIE ==>                                                |     3
        |                                                             |
        |                                DPROP-System:        (name)  |     s
    1   |                                Propagation Request: PR3     |     d*
    2   |                                                             |     d*
    3   |  Enter the member name to be used to store the generated    |     d *
        |  MVG input tables insert job.  The job is stored in data set |     d *
    4   |  "(library)".                                               |     d *
        |                                                             |
    5   |     Member Name    ===> PR3                                 |
        |                                                             |
    S   | Press <ENTER> to continue or enter END command to return.   |
        |                                                             |
    E   |                                                             |
        |_____________________________________________________________|


Enter option or move cursor before desired option.
 Enter END command to return.
```

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Return to panel EKYMP10E to define propagation request PR6.

Panel Flow for PR6: 5

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                     DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR6


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Enter I in the Option field and PR6 in the Propagation Request Name field.

After your input is validated, the following panel is displayed:

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR6

                                          State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 1 to enter the general propagation request specifications. Panel EKYMP20E is displayed.

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------
Command ===>
                                                Propagation Request: PR6

                                                                      More:   - +
  PR Type                   ===> E         ( E, L, F or U                   )
  Mapping Case              ===> 1         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===>           ( ID, DENORM or blank            )
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                       <
    >                                                                       <
    >                                                                       <
    >                                                 <
  User ID                   ===>           ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                       <
    >                                                                       <
    >                                                                       <
    >                                                 <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 616. MVG Input Tables ISPF Application: General Propagation Request Definitions for PR6

Enter END to return to panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR6

                                          State
    1 - Propagation Request      ***  processed  ***
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 2 on panel EKYMP15E to define the DB2 table. The following panel is displayed:

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                Propagation Request: PR6

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===>            ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> T16                 ( a name up to 18 bytes )



         Note: The DB2 table you enter is not checked for existence.







Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 617. MVG Input Tables ISPF Application: DB2 Table for PR6

Enter END to return to panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR6

                                        State
    1 - Propagation Request      ***  processed  ***
    2 - DB2 Tables               ***  processed  ***
    3 - Segments and                 not processed
        Fields/Columns               not processed
    4 - Where Clause (optional)      not processed

    5 - Generate Job                 not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 3 on panel EKYMP15E to define the segments, fields and columns. The following panel is displayed:

```
EKYMP40E --- DPROP MVGIN - Entity Segment for a Mapping Case 1 PR -------------
Command ===>

    F - Fields for the entity segment            Propagation Request: PR6
    P - Fields for the parent segments           Database-
                                                 Organization: HISAM/VSAM

  IMS Database Name         ===> D01       ( a name up to 8 bytes           )
  Segment Name              ===> S16       ( a name up to 8 bytes           )


  PCB Label Name            ===>           ( a name up to 8 bytes           )


  Segment Exit Routine      ===>           ( a name up to 8 bytes, optional )
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760             )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                         )


Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 618. MVG Input Tables ISPF Application: Entity Segment for PR6

To define the fields for the parent segments, enter P on the command line and press ENTER. The following panel is displayed:

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR6
        sequence                             Entity/Internal Segment: S16

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11KEY                              POSITION ===> 1
BYTES    ===> 6         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11K01                              POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C16K01
-------------    3 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11K02                              POSITION ===> 3
BYTES    ===> 4         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C16K02
-------------    4 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D03                              POSITION ===> 7
BYTES    ===> 4         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    5 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D04                              POSITION ===> 11
BYTES    ===> 8         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    6 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D05                              POSITION ===> 19
BYTES    ===> 6         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
```

Figure 619. MVG Input Tables ISPF Application: Parent Segment Fields for PR6 (Part 1 of 2)

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR6
        sequence                             Entity/Internal Segment: S16

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    7 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12D01                              POSITION ===> 1
BYTES    ===> 1         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    8 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12K02                              POSITION ===> 2
BYTES    ===> 4         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C16K03
-------------    9 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12D03                              POSITION ===> 6
BYTES    ===> 20        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------   10 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12D04                              POSITION ===> 26
BYTES    ===> 1         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
```

Figure 620. MVG Input Tables ISPF Application: Parent Segment Fields for PR6 (Part 2 of 2)

Enter END to return to panel EKYMP40E.

To define the fields for the entity segment, enter F on the command line of panel EKYMP40E and press ENTER. The following panel is displayed:

```
EKYMP41E -------- DPROP MVGIN - Entity Segment Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR6
    S - Sort in POSITION sequence                Entity Segment:      S16

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
FLDNAME  ===> F16D01                             POSITION ===> 1
BYTES    ===> 4         DATATYPE ===> F          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C16D04
-------------    2 ------------------------------------------------------------
FLDNAME  ===> F16D02                             POSITION ===> 5
BYTES    ===> 4         DATATYPE ===> F          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C16D05
-------------    3 ------------------------------------------------------------
FLDNAME  ===> F16D03                             POSITION ===> 9
BYTES    ===> 2         DATATYPE ===> H          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C16K06
```

Figure 621. MVG Input Tables ISPF Application: Entity Segment Fields for PR6

Enter END once to return to panel EKYMP40E.

Enter END again to return to panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                 DPROP-System:        (name)
                                                 Propagation Request: PR6

                                        State
    1 - Propagation Request      ***  processed  ***
    2 - DB2 Tables               ***  processed  ***
    3 - Segments and             ***  processed  ***
        Fields/Columns           ***  processed  ***
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. The following panel is displayed:

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

        ________DPROP MVGIN - Generate MVG Input Tables Insert Job____   (name)
       | EKYMPIIE ==>                                                  |   6
       |                                                               |
       |                                DPROP-System:        (name)    |   s
    1  |                                Propagation Request: PR6       |   d *
    2  |                                                               |   d *
    3  |  Enter the member name to be used to store the generated      |
       |  MVG input tables insert job.  The job is stored in data se   |   d *
    4  |  "(library)".                                                 |   d *
       |                                                               |
    5  |   Member Name    ===> PR6                                     |
       |                                                               |
    S  | Press <ENTER> to continue or enter END command to return.     |
       |                                                               |
    E  |                                                               |
       |_______________________________________________________________|


Enter option or move cursor before desired option.
 Enter END command to return.
```

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Return to panel EKYMP10E to define Propagation Request PR7.

Panel Flow for PR7: 5

```
EKYMP10E -------- DPROP MVGIN - Maintain a Propagation Request ----------------
Command ===>

                                                     DPROP-System: (name)

  Enter Option and PR Name:

     Option   ===> I       ( I - Insert a new PR                      )
                           ( U - Update an existing PR                )
                           ( S - Show an existing PR                  )
                           ( D - Delete a PR from the MVG Input Tables )

     PR Name  ===> PR7


  For option "I", you can enter the name of an existing PR to be copied to
  the new PR you want to insert:

     Copy PR  ===>                                 ( ? for PR overview )


Press <ENTER> to continue or enter END command to return.
```

Enter I in the Option field and PR7 in the propagation request Name field.

After your input is validated, the following panel is displayed:

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR7

                                        State
    1 - Propagation Request         not processed
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 1 to enter the general propagation request specifications. Panel EKYMP20E is displayed:

```
EKYMP20E ------------- DPROP MVGIN - Propagation Request ----------------------
Command ===>
                                                Propagation Request: PR7

                                                                      More:   - +
  PR Type                   ===> E         ( E, L, F or U                   )
  Mapping Case              ===> 2         ( 1, 2 or 3                      )
  Mapping Direction         ===> HR        ( HR, RH or TW                   )

  Adding or Replacing a PR  ===> REPL      ( ADD, REPL or blank             )
  Propagation Error Option  ===> BACKOUT   ( BACKOUT, IGNORE or blank       )
  DB2 Table Qualifier       ===>           ( a name up to 8 bytes, optional )
  Number of Error Messages  ===>           ( from 0 up to 32767, optional   )
  PR Set                    ===>           ( a name up to 8 bytes, optional )
  Allow Segment Exit Routines to
  Suppress Propagation      ===>           ( Y, N or blank                  )
  Propagation Exit Routine  ===>           ( only for PR type "U"           )
  Key Sequence in DB2 Table ===>           ( ASC, DSC, ANY or blank         )
  Type of Path Data         ===>           ( ID, DENORM or blank            )
  Avoid unnecessary DB2 Updates
                            ===>           ( Y, N or blank                  )
  Create/Delete Extension Segment
                            ===>           ( Y, N or blank                  )
  Package Bind Options ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <
  User ID                   ===>           ( a name up to 8 bytes, optional )
  Node ID                   ===>           ( a name up to 8 bytes, optional )
  Propagation Request Description ( up to 254 bytes, optional )
  ==>                                                                        <
    >                                                                        <
    >                                                                        <
    >                                                <

Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 622. MVG Input Tables ISPF Application: General Propagation Request Definitions for PR7

Enter END to return to panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR7

                                          State
    1 - Propagation Request      ***  processed  ***
    2 - DB2 Tables                  not processed
    3 - Segments and                not processed
        Fields/Columns              not processed
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 2 on panel EKYMP15E to define the DB2 table. The following panel is displayed:

```
EKYMP30E ----------------- DPROP MVGIN - DB2 Table ---------------------------
Command ===>

                                                Propagation Request: PR7

   DB2 table definition for PR type E, F, and L.


     DB2 Table Qualifier ===>            ( a name up to 8 bytes, optional )
     DB2 Table Name      ===> T17                 ( a name up to 18 bytes )



         Note: The DB2 table you enter is not checked for existence.





Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 623. MVG Input Tables ISPF Application: DB2 Table for PR7

Enter END to return to panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR7

                                        State
    1 - Propagation Request      ***  processed  ***
    2 - DB2 Tables               ***  processed  ***
    3 - Segments and                 not processed
        Fields/Columns               not processed
    4 - Where Clause (optional)      not processed

    5 - Generate Job                 not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 3 on panel EKYMP15E to define the segments, fields and columns. The following panel is displayed:

```
EKYMP50E - DPROP MVGIN - Segments and Fields for a Mapping Case 2 PR ----------
Command ===>


                                                Propagation Request: PR7

                                        State
    1 - Entiy Segment                not processed

    2 - Extension Segments and       not processed
        Fields/Columns               not processed


    E - Return






Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 1 to define the entity segment. The following panel is displayed:

```
EKYMP51E --- DPROP MVGIN - Entity Segment for a Mapping Case 2 PR -------------
Command ===>

                                                  Propagation Request: PR7
                                                  Database-
                                                  Organization: HISAM/VSAM

  IMS Database Name         ===> D01       ( a name up to 8 bytes           )
  Segment Name              ===> S17       ( a name up to 8 bytes           )


  PCB Label Name            ===>           ( a name up to 8 bytes           )


  Segment Exit Routine      ===>           ( a name up to 8 bytes, optional )
  Length of the Segment processed by the Segment Exit Routine
                            ===>           ( from 1 up to 32760             )
  Format of the Segment processed by the Segment Exit Routine
                            ===>           ( F or V                         )


Press <ENTER> to validate your input.  Enter END command to return.
```

Figure 624. MVG Input Tables ISPF Application: Entity Segment for PR7

Enter END to return to panel EKYMP50E.

```
EKYMP50E - DPROP MVGIN - Segments and Fields for a Mapping Case 2 PR ----------
Command ===>


                                                  Propagation Request: PR7

                                         State
    1 - Entity Segment           ***  processed  ***

    2 - Extension Segments and      not processed
        Fields/Columns              not processed


    E - Return





Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 2 to define the extension segments and the fields and columns. The following panel is displayed:

```
EKYMP52E ------------- DPROP MVGIN - Extension Segments -----------------------
Command ===>                                                  SCROLL ===> HALF

    F - Fields for the entity and ext. segments  Propagation Request: PR7
    P - Fields for the parent segments           Entity Segment:      S17

  - Enter an X in the X column below to define an extension segment
  - Remove the X from the X column to unselect an extension segment

Press <ENTER> to validate your input.  Enter END command to return.
_____________________________________________________________________________
X  Seg Name  Seg Exit  Length of Segment processed  Format of Segment processed
                       by the Segment Exit Routine  by the Segment Exit Routine
                       ( from 1 up to 32760 )       ( F or V )
_____________________________________________________________________________
-----------------------------------------------------------------------------  X  S18
-----------------------------------------------------------------------------
****************************** BOTTOM OF DATA ********************************
```

Figure 625. MVG Input Tables ISPF Application: Extension Segments for PR7

To define the fields for the parent segments, enter P on the command line and press ENTER. The following panel is displayed:

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR7
        sequence                             Entity/Internal Segment: S17

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11KEY                              POSITION ===> 1
BYTES    ===> 6         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11K01                              POSITION ===> 1
BYTES    ===> 2         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C17K01
-------------    3 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11K02                              POSITION ===> 3
BYTES    ===> 4         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C17K02
-------------    4 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D03                              POSITION ===> 7
BYTES    ===> 4         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    5 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D04                              POSITION ===> 11
BYTES    ===> 8         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    6 ------------------------------------------------------------
SEGNAME  ===> S11         Segment Level:
FLDNAME  ===> F11D05                              POSITION ===> 19
BYTES    ===> 6         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
```

Figure 626. MVG Input Tables ISPF Application: Parent Segment Fields for PR7 (Part 1 of 2)

```
EKYMPPFE ------- DPROP MVGIN - Parent Segments Fields/Columns -----------------
Command ===>                                                  SCROLL ===> HALF

    S - Sort in Segment Level, POSITION      Propagation Request:     PR7
        sequence                             Entity/Internal Segment: S17

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    7 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12D01                              POSITION ===> 1
BYTES    ===> 1         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    8 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12K02                              POSITION ===> 2
BYTES    ===> 4         DATATYPE ===> P          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C17K03
-------------    9 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12D03                              POSITION ===> 6
BYTES    ===> 20        DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------   10 ------------------------------------------------------------
SEGNAME  ===> S12         Segment Level:
FLDNAME  ===> F12D04                              POSITION ===> 26
BYTES    ===> 1         DATATYPE ===> C          DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
```

Figure 627. MVG Input Tables ISPF Application: Parent Segment Fields for PR7 (Part 2 of 2)

Enter END to return to panel EKYMP52E.

To define the fields for the entity segment, enter F on the command line of panel EKYMP52E and press ENTER. The following panel is displayed:

```
EKYMP53E  DPROP MVGIN - Entity and Extension Segments Fields/Columns ----------    NOTE
Command ===>                                                   SCROLL ===> HALF

    I - Import data structure                    Propagation Request: PR7
    S - Sort in SEGNAME, POSITION sequence       Entity Segment:      S17

   To delete a field, blank out the FLDNAME field.

Press <ENTER> to validate your input.  Enter END command to return.

-------------    1 ------------------------------------------------------------
SEGNAME  ===> S17         Segment Level:
FLDNAME  ===> F17K01                           POSITION ===> 1
BYTES    ===> 20        DATATYPE ===> C        DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C17K04
-------------    2 ------------------------------------------------------------
SEGNAME  ===> S17         Segment Level:
FLDNAME  ===> F17D02                           POSITION ===> 21
BYTES    ===> 3         DATATYPE ===> P        DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C17D05
-------------    3 ------------------------------------------------------------
SEGNAME  ===> S17         Segment Level:
FLDNAME  ===> F17D03                           POSITION ===> 24
BYTES    ===> 1         DATATYPE ===> P        DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C17D06
-------------    4 ------------------------------------------------------------
SEGNAME  ===> S17         Segment Level:
FLDNAME  ===> F17D04                           POSITION ===> 25                 1
BYTES    ===> 2         DATATYPE ===> H        DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===>
-------------    5 ------------------------------------------------------------
SEGNAME  ===> S17         Segment Level:
FLDNAME  ===> F17D05                           POSITION ===> 27                 2
BYTES    ===> 28        DATATYPE ===> VC       DATATYP2 ===>
SCALE    ===>           LENFIELD ===> F17D04
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C17D07
-------------    6 ------------------------------------------------------------
SEGNAME  ===> S18         Segment Level:
FLDNAME  ===> F18D01                           POSITION ===> 1
BYTES    ===> 50        DATATYPE ===> C        DATATYP2 ===>
SCALE    ===>           LENFIELD ===>
FLDEXIT  ===>           FLDETYPE ===>
FLDEBYTE ===>           FLDESCAL ===>
COLNAME  ===> C17D08
```

Figure 628. MVG Input Tables ISPF Application: Entity and Extension Segment Fields for PR7

Notes:

1. F17D04 is the length field of variable-length field F17D05 and therefore has no target DB2 column. The COLNAME field is set to blank.
2. F17D05 is a variable-length character field. Its length field is F17D04. The example specifies, in the LENFIELD field, that F17D04 is the length field of F17D05.

Enter END once to return to panel EKYMP52E.

Enter END again to return to panel EKYMP50E.

```
EKYMP50E - DPROP MVGIN - Segments and Fields for a Mapping Case 2 PR ----------
Command ===>


                                                Propagation Request: PR7

                                        State
    1 - Entity Segment           ***  processed  ***

    2 - Extension Segments and   ***  processed  ***
        Fields/Columns           ***  processed  ***


    E - Return





Enter option or move cursor before desired option.
Enter END command to return.
```

Enter END to return to panel EKYMP15E.

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

                                                DPROP-System:        (name)
                                                Propagation Request: PR7

                                        State
    1 - Propagation Request      ***  processed  ***
    2 - DB2 Tables               ***  processed  ***
    3 - Segments and             ***  processed  ***
        Fields/Columns           ***  processed  ***
    4 - Where Clause (optional)     not processed

    5 - Generate Job                not processed

    S - Show Propagation Request

    E - Return


Enter option or move cursor before desired option.
Enter END command to return.
```

Select option 5 on panel EKYMP15E to generate the MVG input tables insert job. The following panel is displayed:

```
EKYMP15E -- DPROP MVGIN - Propagation Request Definition Selection -----------
Command ===>

         _________DPROP MVGIN - Generate MVG Input Tables Insert Job____(name)
        | EKYMPIIE ==>                                                  | 7
        |                                                               |
        |                                  DPROP-System:        (name)  | s
    1   |                                  Propagation Request: PR7     | d *
    2   |                                                               | d *
    3   |  Enter the member name to be used to store the generated      | d *
        |  MVG input tables insert job.  The job is stored in data set  | d *
    4   |  "(library)".                                                 | d *
        |                                                               |
    5   |     Member Name    ===> PR7                                   |
        |                                                               |
    S   | Press <ENTER> to continue or enter END command to return.     |
        |                                                               |
    E   |                                                               |
        |_______________________________________________________________|


Enter option or move cursor before desired option.
 Enter END command to return.
```

The application creates the insert job and you are placed in ISPF EDIT mode, where you can submit the job.

Providing MVGU control statements: 5

Figure 629 shows the IMS DPROP control statements that tell the MVGU to create the propagation requests in the IMS DPROP directory based on data stored in the MVG input tables.

```
CREATE PR=PR1
CREATE PR=PR2;
CREATE PR=PR3;
CREATE PR=PR6;
CREATE PR=PR7;
```

Figure 629. MVGU control statements to Create the Propagation Requests

Defining Propagation Requests in the MVG Input Tables with SQL Insert Statements: 5

Figure 630 on page 758 through Figure 634 on page 766 describe SPUFI statements that define propagation requests in the MVG input tables. Typically, a user program populates the MVG input tables based on definitions stored in a dictionary.

```
INSERT INTO T096606.DPRIPR
                   (PRID, USERID, PRTYPE, MAPCASE, MAPDIR,  ERROPT  ,
                    ACTION)
       VALUES      ('PR1', ' '  , 'E'   , '1'    ,  'HR' , 'BACKOUT',
                   'REPL');

INSERT INTO T096606.DPRITAB
                   (PRID ,TABQUAL, TABNAME)
       VALUES      ('PR1',' ','T11');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR1','D01','S11', 'E',' ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR1','D01','S11','F11KEY',01,'C ',
                    06,0,'      ',' ','T11','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR1','D01','S11','F11K01',01,'C ',
                    02,0,'      ',' ','T11','C11K01');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR1','D01','S11','F11K02',03,'P ',
                    04,0,'      ',' ','T11','C11K02');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR1','D01','S11','F11D03',07,'C ',
                    04,0,'      ',' ','T11','C11D03');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR1','D01','S11','F11D04',11,'C ',
                    08,0,'      ',' ','T11','C11D04');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR1','D01','S11','F11D05',19,'P ',
                    06,0,'      ',' ','T11','C11D05');
```

Figure 630. SPUFI Statements to Define PR1 in the MVG Input Tables

```
INSERT INTO T096606.DPRIPR                    (PRID, USERID, PRTYPE, MAPCASE, MAPDIR,  ERROPT  ,
                    ACTION)
       VALUES      ('PR2', ' '  , 'E'   , '1'    ,  'HR' , 'BACKOUT',
                   'REPL');

INSERT INTO T096606.DPRITAB
                   (PRID ,TABQUAL, TABNAME)
       VALUES      ('PR2',' ','T12');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR2','D01','S11', 'P',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR2','D01','S12', 'E',' ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR2','D01','S11','F11KEY',01,'C ',
                    06,0,'      ',' ','T12','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR2','D01','S11','F11K01',01,'C ',
                    02,0,'      ',' ','T12','C12K01');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR2','D01','S11','F11K02',03,'P ',
                    04,0,'      ',' ','T12','C12K02');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR2','D01','S11','F11D03',07,'C ',
                    04,0,'      ',' ','T12','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR2','D01','S11','F11D04',11,'C ',
                    08,0,'      ',' ','T12','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR2','D01','S11','F11D05',19,'P ',
                    06,0,'      ',' ','T12','      ');INSERT INTO T096606.DPRIFLD
                    (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR2','D01','S12','F12D01',01,'C ',
                    01,0,'      ',' ','T12','C12D03');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR2','D01','S12','F12K02',02,'P ',
                    04,0,'      ',' ','T12','C12K04');
```

Figure 631. SPUFI Statements to Define PR2 in the MVG Input Tables (Part 1 of 2)

```
INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR2','D01','S12','F12D03',06,'C ',
                    20,0,'      ',' ','T12','C12D05');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR2','D01','S12','F12D04',26,'C ',
                    01,0,'      ',' ','T12','C12D06');
```

Figure 631. SPUFI Statements to Define PR2 in the MVG Input Tables (Part 2 of 2)

```
INSERT INTO T096606.DPRIPR                    (PRID, USERID, PRTYPE, MAPCASE, MAPDIR,  ERROPT  ,
                    ACTION)
       VALUES      ('PR3', ' '  , 'E'   , '2'    ,  'HR' , 'BACKOUT',
                   'REPL');

INSERT INTO T096606.DPRITAB
                   (PRID ,TABQUAL, TABNAME)
       VALUES      ('PR3',' ','T13');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR3','D01','S11', 'P',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR3','D01','S12', 'P',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR3','D01','S13', 'E',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR3','D01','S14', 'X',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR3','D01','S15', 'X',' ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S11','F11KEY',01,'C ',
                    06,0,'      ',' ','T13','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S11','F11K01',01,'C ',
                    02,0,'      ',' ','T13','C13K01');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S11','F11K02',03,'P ',
                    04,0,'      ',' ','T13','C13K02');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S11','F11D03',07,'C ',
                    04,0,'      ',' ','T13','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S11','F11D04',11,'C ',
                    08,0,'      ',' ','T13','      ');
```

Figure 632. SPUFI Statements to Define PR3 in the MVG Input Tables (Part 1 of 3)

```
INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S11','F11D05',19,'P ',
                    06,0,'      ',' ','T13','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S12','F12D01',01,'C ',
                    01,0,'      ',' ','T13','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S12','F12K02',02,'P ',
                    04,0,'      ',' ','T13','C13K03');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S12','F12D03',06,'C ',
                    20,0,'      ',' ','T13','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S12','F12D04',26,'C ',
                    01,0,'      ',' ','T13','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S13','F13D01',01,'H ',
                    02,0,'      ',' ','T13','C13D04');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S13','F13K02',03,'P ',
                    03,0,'      ',' ','T13','C13K05');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S13','F13K03',06,'P ',
                    05,0,'      ',' ','T13','C13K06');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR3','D01','S13','F13D04',11,'H ',
                    02,0,'      ',' ','T13','C13D07');
```

Figure 632. SPUFI Statements to Define PR3 in the MVG Input Tables (Part 2 of 3)

```
INSERT INTO T096606.DPRIFLD
 (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
 BYTES, SCALE, LENFIELD,
 TABQUAL, TABNAME, COLNAME)
 VALUES ('PR3','D01','S13','F13D05',13,'F ',
 04,0,' ',' ','T13','C13D08');

INSERT INTO T096606.DPRIFLD
 (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
 BYTES, SCALE, LENFIELD,
 TABQUAL, TABNAME, COLNAME)
 VALUES ('PR3','D01','S13','F13D06',17,'H ',
 02,0,' ',' ','T13','C13D09');

INSERT INTO T096606.DPRIFLD
 (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
 BYTES, SCALE, LENFIELD,
 TABQUAL, TABNAME, COLNAME)
 VALUES ('PR3','D01','S14','F14D01',01,'C ',
 02,0,' ',' ','T13','C13D10');

INSERT INTO T096606.DPRIFLD
 (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
 BYTES, SCALE, LENFIELD,
 TABQUAL, TABNAME, COLNAME)
 VALUES ('PR3','D01','S14','F14D02',03,'C ',
 05,0,' ',' ','T13','C13D11');

INSERT INTO T096606.DPRIFLD
 (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
 BYTES, SCALE, LENFIELD,
 TABQUAL, TABNAME, COLNAME)
 VALUES ('PR3','D01','S14','F14D03',08,'C ',
 08,0,' ',' ','T13','C13D12');

INSERT INTO T096606.DPRIFLD
 (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
 BYTES, SCALE, LENFIELD,
 TABQUAL, TABNAME, COLNAME)
 VALUES ('PR3','D01','S15','F15D01',01,'P ',
 02,0,' ',' ','T13','C13D13');

INSERT INTO T096606.DPRIFLD
 (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
 BYTES, SCALE, LENFIELD,
 TABQUAL, TABNAME, COLNAME)
 VALUES ('PR3','D01','S15','F15D02',03,'C ',
 30,0,' ',' ','T13','C13D14');
```

Figure 632. SPUFI Statements to Define PR3 in the MVG Input Tables (Part 3 of 3)

```
INSERT INTO T096606.DPRIPR
                   (PRID, USERID, PRTYPE, MAPCASE, MAPDIR,  ERROPT  ,
                    ACTION)
       VALUES      ('PR6', ' '  , 'E'   , '1'    ,  'HR' , 'BACKOUT',
                   'REPL');

INSERT INTO T096606.DPRITAB
                   (PRID ,TABQUAL, TABNAME)
       VALUES      ('PR6',' ','T16');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR6','D01','S11', 'P',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR6','D01','S12', 'P',' ');

INSERT INTO T096606.DPRISEG
                   (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES      ('PR6','D01','S16', 'E',' ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S11','F11KEY',01,'C ',
                    06,0,'      ',' ','T16','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S11','F11K01',01,'C ',
                    02,0,'      ',' ','T16','C16K01');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S11','F11K02',03,'P ',
                    04,0,'      ',' ','T16','C16K02');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S11','F11D03',07,'C ',
                    04,0,'      ',' ','T16','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S11','F11D04',11,'C ',
                    08,0,'      ',' ','T16','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S11','F11D05',19,'P ',
                    06,0,'      ',' ','T16','      ');
```

Figure 633. SPUFI Statements to Define PR6 in the MVG Input Tables (Part 1 of 2)

```
INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S12','F12D01',01,'C ',
                    01,0,'      ',' ','T16','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S12','F12K02',02,'P ',
                    04,0,'      ',' ','T16','C16K03');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S12','F12D03',06,'C ',
                    20,0,'      ',' ','T16','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S12','F12D04',26,'C ',
                    01,0,'      ',' ','T16','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S16','F16D01',01,'F ',
                    04,0,'      ',' ','T16','C16D04');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S16','F16D02',05,'F ',
                    04,0,'      ',' ','T16','C16D05');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR6','D01','S16','F16D03',09,'H ',
                    02,0,'      ',' ','T16','C16K06');
```

Figure 633. SPUFI Statements to Define PR6 in the MVG Input Tables (Part 2 of 2)

```
INSERT INTO T096606.DPRIPR
                  (PRID, USERID, PRTYPE, MAPCASE, MAPDIR,  ERROPT  ,
                   ACTION)
       VALUES     ('PR7', ' '  , 'E'   , '2'    ,  'HR' , 'BACKOUT',
                  'REPL');

INSERT INTO T096606.DPRITAB
                  (PRID ,TABQUAL, TABNAME)
       VALUES     ('PR7',' ','T17');

INSERT INTO T096606.DPRISEG
                  (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES     ('PR7','D01','S11', 'P',' ');

INSERT INTO T096606.DPRISEG
                  (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES     ('PR7','D01','S12', 'P',' ');

INSERT INTO T096606.DPRISEG
                  (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES     ('PR7','D01','S17', 'E',' ');

INSERT INTO T096606.DPRISEG
                  (PRID, DBNAME, SEGNAME, ROLE, FORMAT)
       VALUES     ('PR7','D01','S18', 'X',' '); INSERT INTO T096606.DPRIFLD
(PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                   BYTES, SCALE, LENFIELD,
                   TABQUAL, TABNAME, COLNAME)
       VALUES     ('PR7','D01','S11','F11KEY',01,'C ',
                   06,0,'      ',' ','T17','      ');

INSERT INTO T096606.DPRIFLD
                  (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                   BYTES, SCALE, LENFIELD,
                   TABQUAL, TABNAME, COLNAME)
       VALUES     ('PR7','D01','S11','F11K01',01,'C ',
                   02,0,'      ',' ','T17','C17K01');

INSERT INTO T096606.DPRIFLD
                  (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                   BYTES, SCALE, LENFIELD,
                   TABQUAL, TABNAME, COLNAME)
       VALUES     ('PR7','D01','S11','F11K02',03,'P ',
                   04,0,'      ',' ','T17','C17K02');

INSERT INTO T096606.DPRIFLD
                  (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                   BYTES, SCALE, LENFIELD,
                   TABQUAL, TABNAME, COLNAME)
       VALUES     ('PR7','D01','S11','F11D03',07,'C ',
                   04,0,'      ',' ','T17','      '); INSERT INTO T096606.DPRIFLD
(PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                   BYTES, SCALE, LENFIELD,
                   TABQUAL, TABNAME, COLNAME)
       VALUES     ('PR7','D01','S11','F11D04',11,'C ',
                   08,0,'      ',' ','T17','      ');

INSERT INTO T096606.DPRIFLD
                  (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                   BYTES, SCALE, LENFIELD,
                   TABQUAL, TABNAME, COLNAME)
       VALUES     ('PR7','D01','S11','F11D05',19,'P ',
                   06,0,'      ',' ','T17','      ');

INSERT INTO T096606.DPRIFLD
                  (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                   BYTES, SCALE, LENFIELD,
                   TABQUAL, TABNAME, COLNAME)
       VALUES     ('PR7','D01','S12','F12D01',01,'C ',
                   01,0,'      ',' ','T17','      ');

INSERT INTO T096606.DPRIFLD
                  (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                   BYTES, SCALE, LENFIELD,
                   TABQUAL, TABNAME, COLNAME)
       VALUES     ('PR7','D01','S12','F12K02',02,'P ',
                   04,0,'      ',' ','T17','C17K03'); INSERT INTO T096606.DPRIFLD
(PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                   BYTES, SCALE, LENFIELD,
                   TABQUAL, TABNAME, COLNAME)
       VALUES     ('PR7','D01','S12','F12D03',06,'C ',
                   20,0,'      ',' ','T17','      ');
```

Figure 634. SPUFI Statements to Define PR7 in the MVG Input Tables (Part 1 of 2)

```
INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR7','D01','S12','F12D04',26,'C ',
                    01,0,'      ',' ','T17','      ');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR7','D01','S17','F17K01',01,'C ',
                    20,0,'      ',' ','T17','C17K04');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR7','D01','S17','F17D02',21,'P ',
                    03,0,'      ',' ','T17','C17D05'); INSERT INTO T096606.DPRIFLD
(PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR7','D01','S17','F17D03',24,'P ',
                    01,0,'      ',' ','T17','C17D06');

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR7','D01','S17','F17D04',25,'H ',
                    02,0,'      ',' ','T17','      ');            1

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR7','D01','S17','F17D05',27,'VC',
                    28,0,'F17D04',' ','T17','C17D07');            2

INSERT INTO T096606.DPRIFLD
                   (PRID, DBNAME, SEGNAME, FLDNAME, POSITION, DATATYPE,
                    BYTES, SCALE, LENFIELD,
                    TABQUAL, TABNAME, COLNAME)
       VALUES      ('PR7','D01','S18','F18D01',01,'C ',
                    50,0,'      ',' ','T17','C17D08');
```

Figure 634. SPUFI Statements to Define PR7 in the MVG Input Tables (Part 2 of 2)

Notes:

1. F17D04 is the length field of variable-length field F17D05 and therefore has no target DB2 column. The COLNAME column of the DPRIFLD table is set to blanks.
2. F17D05 is a variable-length character field. Its length field is F17D04. The example specifies, in the LENFIELD column of the DPRIFLD table, that F17D04 is the length field of F17D05.

Providing MVGU control statements: 5

Figure 635 on page 768 shows the IMS DPROP control statements that tell the MVGU to create the propagation requests in the IMS DPROP directory based on data stored in the MVG input tables.

```
CREATE PR=PR1
CREATE PR=PR2;
CREATE PR=PR3;
CREATE PR=PR6;
CREATE PR=PR7;
```

Figure 635. MVGU control statements to Create the Propagation Requests

Example 6: Propagating to DSI Tables

To describe different propagation possibilities, the database shown in Figure 636 serves as our model for the examples of this topic.

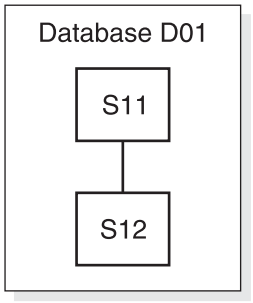

Figure 636. Model Database for Example 6

This example describes:

- How to propagate S11 to a condensed DSI table with a PR belonging to mapping case 1.
- How to propagate S12 to a non-condensed DSI table with a PR belonging to mapping case 1.

Visual Overview of the Performed Propagation: 6

Figure 637 illustrates the propagation of the segments of our model IMS database D01 to a condensed DSI table and a non-condensed DSI table. The propagation is done with propagation requests belonging to mapping case 1.

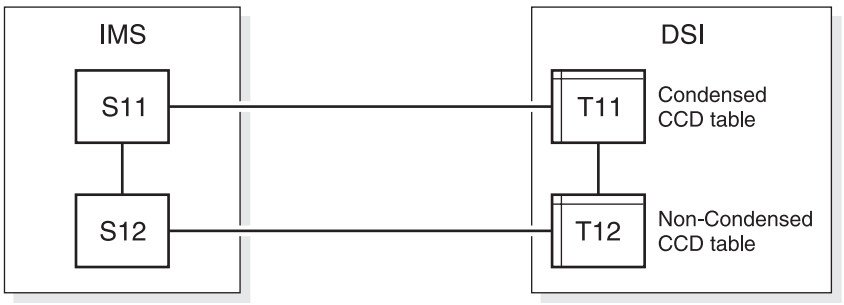

Figure 637. Overview of Propagation of Database D01

Figure 638 on page 769 illustrates that:

- Segment S11 is propagated by PR1 to table T11
- Segment S12 is propagated by PR2 to table T12

Figure 638 is an overview of how the IMS fully concatenated key is mapped to the DB2 primary key for the condensed DSI table. The non-condensed DSI table has no primary key. The figure shows how IMS fields belonging to the fully concatenated key of the entity segments are mapped to columns of the DB2 primary key for the condensed table.

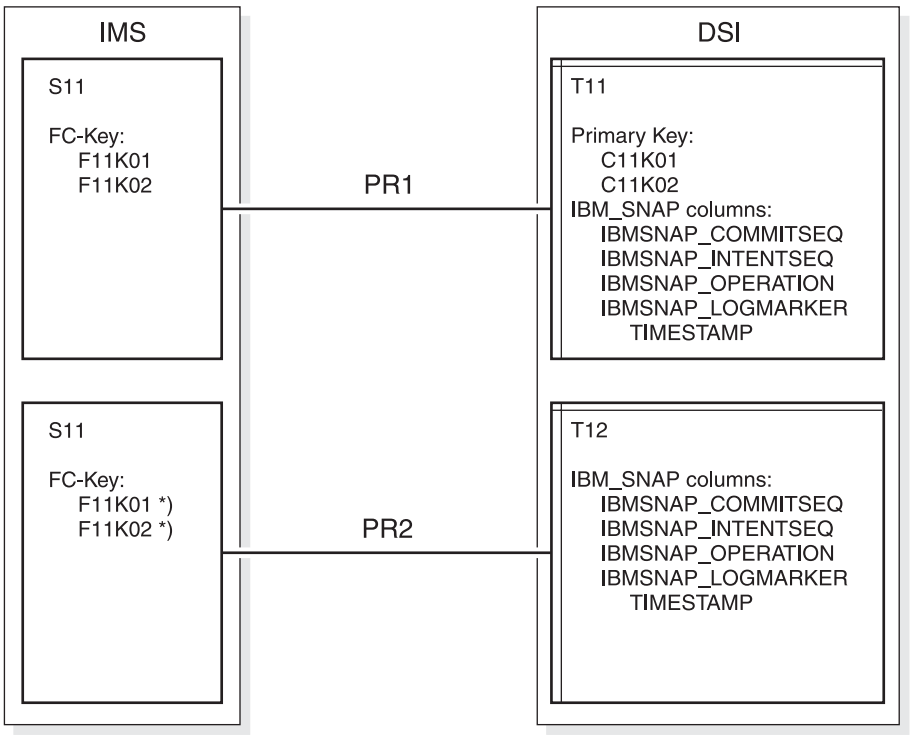

Figure 638. Mapping of the Keys

Field Mapping Overview: 6

Table 57 on page 770 and Table 58 on page 770 show the characteristics of the source IMS fields and the DSI DB2 columns. The tables contain the following information:

- For the IMS source field:
  - The name of the IMS field (as defined to IMS DPROP either on the NAME keyword of the DataRefresher FIELD statement, or in the FLDNAME column of the DPRIFLD MVG input table)
  - The data type (as defined to IMS DPROP either on the TYPE keyword of the DataRefresher FIELD statement, or in the DATATYPE column of the DPRIFLD MVG input table)
  - The length in bytes (as defined to IMS DPROP either on the BYTES keyword of the DataRefresher FIELD statement or in the BYTES column of the DPRIFLD MVG input table)
- For the DB2 DSI column:
  - The name of the DSI DB2 column (as defined to DB2 in the CREATE TABLE statement)

– The data type of the DSI DB2 column (as defined to DB2 in the CREATE TABLE statement)

For a decimal column, specify the number of digits of the column in the CREATE TABLE statement. However, for a packed IMS field, do not specify the number of digits; instead, specify the number of bytes.

Table 57. Source Segment: S11 - DSI Table: T11

Source IMS field			DSI DB2 column	
Field Name	**Data Type**	**Length (in bytes)**	**Column Name**	**Data Type**
-	-	-	IBMSNAP_COMMITSEQ	CHAR(10)
-	-	-	IBMSNAP_INTENTSEQ	CHAR(10)
-	-	-	IBMSNAP_OPERATION	CHAR(1)
-	-	-	IBMSNAP_LOGMARKER	TIMESTAMP
F11K01	C	2	C11K01	CHAR(2)
F11K02	P	4	C11K02	DECIMAL(7)
F11D03	C	4	C11D03	CHAR(4)
F11D04	C	8	C11D04	CHAR(8)
F11D05	P	6	C11D05	DECIMAL(11)

Table 58. Source Segment: S12 (and Its Concatenated Key) - DSI Table: T12

Source IMS field			DSI DB2 column	
Field Name	**Data Type**	**Length (in bytes)**	**Column Name**	**Data Type**
-	-	-	IBMSNAP_COMMITSEQ	CHAR(10)
-	-	-	IBMSNAP_INTENTSEQ	CHAR(10)
-	-	-	IBMSNAP_OPERATION	CHAR(1)
-	-	-	IBMSNAP_LOGMARKER	TIMESTAMP
F11K01	C	2	C12K01	CHAR(2)
F11K02	P	4	C12K02	DECIMAL(7)
F12D01	C	1	C12D03	CHAR(1)
F12K02	P	4	C12K04	DECIMAL(7)
F12D03	C	20	C12D05	CHAR(20)
F12D04	C	1	C12D06	CHAR(1)

DBD Definitions: 6

The DBD control statements for the model database are described in Figure 639 on page 771.

```
DBD     NAME=D01,ACCESS=HISAM, EXIT=(EKYRUP00)
DATASET DD1=DDNAME1,OVFLW=DDNAME2,DEVICE=3380

SEGM    NAME=S11,PARENT=0,BYTES=24
FIELD   NAME=(F11KEY,SEQ,U),TYPE=C,BYTES=6,START=1
FIELD   NAME=F11K01,TYPE=C,BYTES=2,START=1
FIELD   NAME=F11K02,TYPE=P,BYTES=4,START=3
FIELD   NAME=F11D03,TYPE=C,BYTES=4,START=7
FIELD   NAME=F11D04,TYPE=C,BYTES=8,START=11
FIELD   NAME=F11D05,TYPE=P,BYTES=6,START=19

SEGM    NAME=S12,PARENT=S11,BYTES=26
FIELD   NAME=(F12KEY,SEQ,U),TYPE=C,BYTES=4,START=2
FIELD   NAME=F12D01,TYPE=C,BYTES=1,START=1
FIELD   NAME=F12K02,TYPE=P,BYTES=4,START=2
FIELD   NAME=F12D03,TYPE=C,BYTES=20,START=6
FIELD   NAME=F12D04,TYPE=C,BYTES=1,START=26

DBDGEN
FINISH
END
```

Figure 639. DBD control statements for the Model Database (Example 6)

The EXIT keyword in the DBD control statement tells IMS to give control to EKYRUP00, the IMS DPROP-provided Data Capture Exit routine, when a DL/I call updates a segment of this DBD.

Usually, IMS DPROP and DataRefresher do not require you to define the non-key fields in the IMS DBD. They will not use such definitions.

Table Definitions: 6

The SQL statements to define the DSI tables are shown in Figure 640 on page 772 and Figure 641 on page 772.

Notes:

1. The four IBM_SNAP columns must be at the top of the table structure and in the order shown.
2. As table T11 has the four IBM_SNAP columns and a primary key, it is recognizable as a condensed DSI table.
3. As table T12 has the four IBM_SNAP columns and no primary key, it is recognizable as a non-condensed DSI table.

```
CREATE TABLE prefix.T11
          (IBMSNAP_COMMITSEQ CHAR(10) FOR BIT DATA NOT NULL,
           IBMSNAP_INTENTSEQ CHAR(10) FOR BIT DATA NOT NULL,
           IBMSNAP_OPERATION CHAR(1) NOT NULL,
           IBMSNAP_LOGMARKER TIMESTAMP,
           C11K01            CHAR(2)       NOT NULL,
           C11K02            DECIMAL(7)    NOT NULL,
           C11D03            CHAR(4)       NOT NULL WITH DEFAULT,
           C11D04            CHAR(8)       NOT NULL WITH DEFAULT,
           C11D05            DECIMAL(11)   NOT NULL WITH DEFAULT,
       PRIMARY KEY (C11K01,C11K02))
       IN database.tablespace;

CREATE UNIQUE INDEX prefix.TX11
       ON prefix.T11 (C11K01,C11K02);
```

Figure 640. Table T11

```
CREATE TABLE prefix.T12
          (IBMSNAP_COMMITSEQ CHAR(10) FOR BIT DATA NOT NULL,
           IBMSNAP_INTENTSEQ CHAR(10) FOR BIT DATA NOT NULL,
           IBMSNAP_OPERATION CHAR(1) NOT NULL,
           IBMSNAP_LOGMARKER TIMESTAMP,
           C12K01            CHAR(2)       NOT NULL,
           C12K02            DECIMAL(7)    NOT NULL,
           C12D03            CHAR(1)       NOT NULL WITH DEFAULT,
           C12K04            DECIMAL(7)    NOT NULL,
           C12D05            CHAR(20)      NOT NULL WITH DEFAULT,
           C12D06            CHAR(1)       NOT NULL WITH DEFAULT,
       IN database.tablespace;
```

Figure 641. Table T12

Defining Propagation Requests with DataRefresher: 6

Defining the DXTPSB: 6

DataRefresher requires that you describe the IMS database to DataRefresher by providing a CREATE DXTPSB command. The DataRefresher control statements for our model database are shown in Figure 642 on page 773.

```
                                                                    NOTE
CREATE  DXTPSB   NAME=PSB1
        DXTPCB   NAME=PSB1PCB1,DBACCESS=HISAM,DBNAME=D01

       SEGMENT   NAME=S11,PARENT=0,BYTES=24
         FIELD   NAME=F11KEY,TYPE=C,BYTES=6,START=1,SEQFLD=R
         FIELD   NAME=F11K01,TYPE=C,BYTES=2,START=1
         FIELD   NAME=F11K02,TYPE=P,BYTES=4,START=3
         FIELD   NAME=F11D03,TYPE=C,BYTES=4,START=7
         FIELD   NAME=F11D04,TYPE=C,BYTES=8,START=11
         FIELD   NAME=F11D05,TYPE=P,BYTES=6,START=19

       SEGMENT   NAME=S12,PARENT=S11,BYTES=26
         FIELD   NAME=F12D01,TYPE=C,BYTES=1,START=1
         FIELD   NAME=F12K02,TYPE=P,BYTES=4,START=2
         FIELD   NAME=F12D03,TYPE=C,BYTES=20,START=6
         FIELD   NAME=F12D04,TYPE=C,BYTES=1,START=26
```

Figure 642. DataRefresher control statements for the Model Database (Example 6)

Defining the DXTVIEWs: 6

DataRefresher requires a view description for each hierarchical path of the database. The DataRefresher control statements used to define the view for our example database is shown in Figure 643.

```
CREATE  DXTVIEW NAME=VIEWA,
                DXTPSB=PSB1,DXTPCB=PSB1PCB1,SEGMENT=S12,MINSEGM=S11,
                FIELDS=(F11K01,F11K02,F12K02);
```

Figure 643. View for Segments S11 and S12

Providing DataRefresher UIM SUBMIT Commands and EXTRACT Statements: 6

Figure 644 on page 774 and Figure 641 on page 772 illustrate the DataRefresher SUBMIT commands and EXTRACT statements used to define the propagation requests.

The DataRefresher SUBMIT/EXTRACT creates the propagation requests in the IMS DPROP directory but does not load the data to the DB2 tables. This is the responsibility of the DataRefresher DEM. Refer to “Providing DataRefresher DEM control statements: 5 ” on page 720 for examples of the DataRefresher DEM control statement.

Specifying DBS=DSI causes DataRefresher to create data for the IBMSNAP columns, as described in the appropriate Administrators Guide for your propagation mode.

Note: The IBMSNAP columns are not included on the EXTRACT statement.

```
SUBMIT   EXTID=PR1,
               CD=JCS,
               JCS=DDJCS01,
               DBS=DSI,
               FORMAT=SOURCE,
               USERDECK='ENFORCE NO',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,
                         MAPCASE=1,
                         MAPDIR=HR,
                         ERROPT=BACKOUT,
                         ACTION=REPL'

EXTRACT  INTO   prefix.T11 (C11K01 NOT NULL,
                            C11K02 NOT NULL,
                            C11D03 NOT NULL WITH DEFAULT,
                            C11D04 NOT NULL WITH DEFAULT,
                            C11D05 NOT NULL WITH DEFAULT)

         OPTIONS            (FLDERR(HALT))

         SELECT             VIEWA.F11K01,
                            VIEWA.F11K02,
                            VIEWA.F11D03,
                            VIEWA.F11D04,
                            VIEWA.F11D05

         FROM               VIEWA;
```

Figure 644. DataRefresher SUBMIT/EXTRACT for Table T11

```
SUBMIT   EXTID=PR2,
               CD=JCS,
               JCS=DDJCS01,
               DBS=DSI,
               FORMAT=SOURCE,
               USERDECK='ENFORCE NO',
               MAPEXIT=EKYMCE00,
               MAPUPARM='PRTYPE=E,
                         MAPCASE=1,
                         MAPDIR=HR,
                         ERROPT=BACKOUT,
                         ACTION=REPL'

EXTRACT  INTO   prefix.T12 (C12K01 NOT NULL,
                            C12K02 NOT NULL,
                            C12D03 NOT NULL WITH DEFAULT,
                            C12K04 NOT NULL,
                            C12D05 NOT NULL WITH DEFAULT,
                            C12D06 NOT NULL WITH DEFAULT)

         OPTIONS            (FLDERR(HALT))

         SELECT             VIEWA.F11K01,
                            VIEWA.F11K02,
                            VIEWA.F12D01,
                            VIEWA.F12K02,
                            VIEWA.F12D03,
                            VIEWA.F12D04

         FROM               VIEWA;
```

Figure 645. DataRefresher SUBMIT/EXTRACT for Table T12

Providing DataRefresher DEM control statements: 6

Figure 646 shows an example of a DataRefresher DEM control statement that you can use to extract the data from the IMS database.

```
INITDEM NAME=DEM1
USE DXTPSB=PSB1;
```

Figure 646. DataRefresher DEM control statements (Example 6)

Notices

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

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

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

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

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute

these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. All rights reserved.

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

Programming Interface Information

This publication is intended to help you administer IMS DataPropagator, hereafter called IMS DPROP.

This publication also documents general-use programming interface and associated guidance information provided by IMS DPROP.

General-use programming interfaces allow the customer to write programs that obtain the services of IMS DPROP.

General-use programming interface and associated guidance information is identified where it occurs, either by an introductory statement to a topic or by the following marking:

Notice
This topic documents general-use programming interface and associated guidance information.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AD/Cycle
BookManager
DataPropagator
DataRefresher
DB2
DB2 Universal Database
DFS
DXT
IBM
IMS
IMS/ESA
MQSeries
MVS
MVS/DFP
MVS/ESA
MVS/SP
Notes
OS/2
OS/390
QMF
RACF
Redbooks
RUP
SQL/DS
z/OS

Glossary of Terms and Abbreviations

A

abort record. An IMS DataPropagator propagation log record (38*nn* or 5938), indicating that the associated unit of work will not be committed by IMS and should not be propagated to DB2. *Compare with commit record*.

ACB. Application control block. Located in IMS.

ACDC. Asynchronous changed data capture.

Apply Program. A component of IMS MQ-DPROP that reads the MQSeries messages containing the changed data and passes it to the RUP. RUP transforms the changed data into relational format and updates the DB2 target tables.

Archive utility. A utility that filters out propagation log records from the records written to the IMS logs and writes them to Changed Data Capture data sets (CDCDSs).

asynchronous changed data capture. An IMS function that captures the changes needed for IMS DPROP asynchronous propagation and saves them on the IMS logs. The function is mandatory for IMS DPROP asynchronous propagation and is either implemented by an SPE (IMS 3.1) or built into the program (subsequent releases of IMS).

asynchronous propagation. The propagation of data at a later time, not within the same unit of work as the update call.

Audit Extract utility. An IMS DPROP utility that inserts the IMS DPROP audit records written to SMF into the IMS DPROP audit table.

AUDU. Audit Extract utility.

B

Batch Log data set. A data set that an IMS batch job uses to store propagation log records needed for IMS DPROP asynchronous propagation.

C

CAF. Call attach facility.

CCU. Consistency Check utility.

CDCDS. Changed Data Capture data sets.

CDCDS Registration utility. An IMS DPROP asynchronous propagation utility that registers new CDCDS to DBRC.

CDCDS Unregistration utility. An IMS DPROP asynchronous propagation utility that deletes CDCDS entries from DBRC.

CDU. CDCDS Unregistration utility.

CEC. central electronics complex.

Changed Data Capture data set (CDCDS). The data sets that the archive utility uses to store the IMS DPROP asynchronous propagation log records filtered during the archive process. CDCDSs contain only the propagation log records. These log records are used by the Selector in place of the corresponding SLDSs, that contain all IMS changes.

Changed Data Capture exit routine. See DB2 Changed Data Capture exit routine

Changed Data Capture function. See DB2 Changed Data Capture function.

commit record. An IMS DPROP asynchronous propagation log record (9928, 37*nn*, 41*nn*, or 5937) indicating that the associated unit of work has been committed by IMS and should be propagated to DB2. *Compare with abort record*.

concatenated key. See “IMS concatenated key” and “conceptual concatenated key.”

conceptual concatenated key. The conceptual concatenated key of a segment consists of the concatenated keys of the segment’s immediate physical parent and physical ancestors. Unlike the Conceptual *fully* Concatenated key, the conceptual concatenated key does not include the concatenated key of the segment itself.

conceptual fully concatenated key. The conceptual fully concatenated key is an IMS DPROP concept useful for the propagation of entity segments that do not have a unique IMS fully concatenated key; but that are nevertheless uniquely identifiable.

The conceptual fully concatenated key of a segment consists of these parts:
- the concatenated key of the segment
- the concatenated keys of the segment’s physical parent and physical ancestors

The conceptual fully concatenated key is therefore the combination of these parts:
- the IMS fully concatenated key
- the ID fields (if any) of the segment that contribute to the concatenated key of the segment
- the ID fields (if any) of the physical parent or ancestors that contribute to the concatenated keys of the physical parent or ancestor

So, the conceptual fully concatenated key is equal to that hypothetical IMS fully concatenated key, that you would see if including the ID fields into the IMS key-field at each hierarchical level.

The concept of conceptual fully concatenated key allows the support of segments with a unique conceptual fully concatenated key, much in the same way as segments with a unique IMS fully concatenated key.

concatenated key. The concatenated key is an IMS DPROP concept useful for the propagation of entity segments that are neither unique under their parent nor have a unique IMS key, but that are nevertheless uniquely identifiable through ID fields.

The concatenated key is a combination of these fields that identify the segment uniquely under its parent:
- the non-unique IMS key field (if any)
- ID fields

For segments having a unique IMS key field, the conceptual key and the IMS key field are identical.

Consistency Check utility (CCU). An IMS DPROP utility that checks whether the data that has been propagated between IMS and DB2 databases is consistent. If not, it reports the inconsistencies and generates statements the DBA can use to fix the inconsistencies. The CCU is applicable when generalized mapping cases are being used.

containing IMS segment. An IMS segment that contains internal segments (embedded structures) propagated by mapping case 3 Propagation Requests. It is referred to interchangeably as a “containing IMS segment” or “containing segment.”

containing segment. See containing IMS segment.

CRU. CDCDS Registration utility.

D

Data Capture exit routine. See IMS data capture exit routine.

data capture function. An IMS function that captures the changes needed for data propagation.

DataRefresher. An IBM licensed program that lets you extract selected operational data on a periodic or one-time basis.

Data Extract Manager (DEM). A DataRefresher component that extracts the IMS data to which changes will subsequently be propagated. DEM also creates control statements for the DB2 Load utility to load the extracted IMS data into DB2 tables.

data propagation. The application of changes to one set of data to the copy of that data in another database system. See also synchronous propagation and IMS DPROP asynchronous propagation.

DataRefresher DEM. DataRefresher data extract manager.

DataRefresher Map Capture exit routine (MCE). See Map Capture exit routine.

DataRefresher UIM. See User Input Manager.

DBRM. Database Request Module.

DB2 commit count. The number of IMS commit records that the IMS DPROP asynchronous propagation receiver is to apply to DB2 before it issues a DB2 commit.

DB2 Changed Data Capture exit routine. The routine to which the DB2 Changed Data Capture function passes the DB2 changes it has captured for propagation. This routine can be the IMS DPROP HUP routine, that propagates data, or your own exit routine.

DB2 Changed Data Capture function. A DB2 function that captures the DB2 changes needed for data propagation.

DB2 Changed Data Capture subexit routine. An optional IMS DPROP exit routine invoked whenever the HUP is called by DB2 changed data capture. The DB2 Changed Data Capture subexit routine can typically be used to perform generalized functions such as auditing all of the captured DB2 changes.

DB2-to-IMS propagation. Propagation of changed DB2 tables to IMS segments. It can be either:
- One-way DB2-to-IMS propagation
- DB2-to-IMS propagation, as part of two-way propagation

DBD. Database definition. The collection of macroparameter statements that describes an IMS database. These statements describe the hierarchical structure, IMS organization, device type, segment length, sequence fields, and alternate search fields. The statements are assembled to produce database description blocks.

DBDLIB. Database definition library.

DBPCB. Database program communication block.

DEDB. Data entry database.

DEM. Data Extract Manager.

directory. See IMS DPROP directory.

DLU. DL/1 Load Utilities. IMS DPROP utilities that are used to create (or re-create) the IMS databases from the content of the propagated DB2 tables. You can use DLU if you have implemented DB2 to IMS or two-way propagation.

DPROP-NR. The abbreviation for IBM IMS DataPropagator MVS/ESA through Version 2.2. At Version 3.1 the product name changed to IMS DataPropagator, abbreviated as IMS DPROP.

E

EKYMQCAP. The Capture component of MQ-DPROP. EKYMQCAP is an IMS data Capture exit routine. It runs as an extension to the updating IMS application programs, but it is transparent to them. EKYMQCAP obtains the changed data from the IMS Data Capture function and sends this data via MQSeries messages to the Apply Program.

EKYRESLB Dynamic Allocation exit routine. An IMS DPROP exit routine that can be used to allocate dynamically the IMS DPROP load module library to the EKYRESLB DD-name.

entity segment. The data being mapped from IMS to DB2 comes from one single hierarchic path down to a particular segment. This segment is called the entity segment. See also mapping case 1.

ER. Extract request.

Event Marker. A component of MQ-DPROP that runs on the same system as the IMS source databases. It is used to identify an event that occurs on the Source System. The customer must execute the Event Marker on the Source System at the time that the event occurs.

The Event Marker transmits an MQSeries message that identifies the event to the Apply Program. This MQSeries message is transmitted in FIFO sequence and in the same Propagation Data Streams as the changed IMS data.

When an occurrence of the Apply Program processes this message, the content of the target DB2 tables of this occurrence of the Apply Program reflect the content of the IMS source databases at the time that the Event Marker was executed on the Source System.

The Event Marker is used for an automated stop of the Apply Program when the content of the target DB2 tables reflects a particular Source System point in time.

exit routines. IMS DPROP contains seven exit routines. See the individual glossary entries for:
- DB2 Changed Data Capture exit routine
- DB2 Changed Data Capture subexit routine
- IMS Data Capture exit routine
- Field exit routine
- Map Capture exit routine
- Propagation exit routine
- Segment exit routine
- User exit routine

extension segment. The data being mapped from IMS to DB2 comes from a single hierarchic path down to an entity segment and from any segments immediately subordinate to the entity segment. The segments subordinate to the entity segment can have zero or one occurrence beneath a single occurrence of the entity segment. This type of subordinate segment is called an extension segment (as it extends the data in the entity segment). See also mapping case 2.

extract request (ER). A DataRefresher request to extract IMS data. Extract requests become IMS DPROP propagation requests once they are validated by the IMS DPROP MCE.

F

Field exit routine. An IMS DPROP exit routine you can write to complement the logic of IMS DPROP's generalized mapping cases. Field exit routines are typically used to convert an individual IMS data field between a customer format IMS DPROP does not support and a format you have defined in your propagation request.

FIFO. First-In-First-Out

fully concatenated key. See IMS fully concatenated key and conceptual fully concatenated key.

G

generalized mapping cases. The mapping cases provided by IMS DPROP. See mapping case 1, mapping case 2 and mapping case 3.

group definition file. The file that the Group Unload utility (GUU) uses to store the IMS sources that it extracts from the IMS DPROP directory tables. *See also, SCF Compare job and SCF Apply job*.

Group Unload utility (GUU). The IMS DPROP asynchronous propagation utility that extracts details of all IMS sources for the specified propagation group from the IMS DPROP directory tables at the receiver site and writes them to the Group Definitions File. *See also, SCF Compare job and SCF Apply job*.

GUU. Group Unload utility.

H

hierarchical update program (HUP). The IMS DPROP component that does the actual DB2-to-IMS propagation. HUP is the IMS DPROP-provided DB2 Changed Data Capture exit routine. The DB2 Changed Data Capture function calls HUP and provides to HUP the changed IMS rows.

Hierarchical to Relational propagation. This is one-way hierarchical to relational propagation: the one-way propagation of changed IMS segments to DB2 tables. The terms *hierarchical to relational propagation* and *one-way IMS-to-DB2 propagation* are interchangeable.

HUP. Hierarchical Update program.

HSSR. High speed sequential retrieval.

I

ID fields. *Identification (ID) fields* are non-key fields that:
- uniquely identify a segment under its parent
- do not change their value

Typical examples of IMS segments with ID fields, are segments where the data base administrator has not defined the ID fields as part of the IMS Key field. For example because the IMS applications need to retrieve the segment in another sequence than the ascending sequence of the ID fields.

identification fields. See ID fields.

IMS concatenated key. For an IMS segment, the concatenated key consists of:
- The key of the segment's immediate parent, and
- The keys of the segment's ancestors

Unlike the IMS **fully** concatenated key of the segment, the concatenated key does not include the key of the segment itself.

A logical child segment has two concatenated keys: a physical concatenated key and a logical concatenated key. The physical concatenated key consists of the key of the segment's physical parent and the keys of the physical ancestors of the physical parent. The logical concatenated key consists of the key of the segment's logical parent and the keys of the physical ancestors of the logical parent.

IMS Data Capture exit routine. The routine to which the IMS Data Capture function passes the IMS changes it has captured for propagation. For synchronous propagation, this routine can be the IMS DPROP RUP routine, that propagates data, or your own exit routine. For IMS DPROP asynchronous propagation, the data capture exit routine is a program you write that gets the changed data from IMS. Other programs that you write will later invoke IMS DPROP with the changed IMS data.

IMS data capture function. An IMS function that captures the changes needed for data propagation.

IMS DPROP. The abbreviated name for the IBM IMS DataPropagator product. Previously, this product was called IMS DataPropagator, abbreviated as DPROP-NR.

IMS DPROP directory. A set of DB2 tables containing the mapping and control information necessary to perform propagation.

IMS fully concatenated key. For an IMS segment, the fully concatenated key consists of:
- The key of the segment,
- The key of the segment's immediate parent, and
- The keys of the segment's ancestors.

Unlike the IMS concatenated key of the segment, the fully concatenated key includes the key of the segment itself.

IMS INQY data. The first 9904 (update) record in each IMS unit of work (UOW) contains IMS INQY data (transaction name, PSB name, and user ID). This information is written to the PRDS for the propagation group as the first record of the UOW.

IMS log files. The files that IMS uses to store details of all changes to IMS data. See also, batch log data sets, online data sets (OLDSs), system log data sets (SLDSs), and Changed Data Capture data sets (CDCDSs).

IMS logical concatenated key. One of the two IMS concatenated keys of a logical child segment (the other is an IMS physical concatenated key). The logical concatenated key consists of:
- The key of the segment's logical parent, and
- The keys of the physical ancestors of the logical parent.

IMS physical concatenated key. One of the two IMS concatenated keys of a logical child segment (the other is an IMS logical concatenated key). The physical concatenated key consists of:
- The key of the segment's physical parent, and
- The keys of the physical ancestors of the physical parent.

IMS-to-DB2 propagation. This is the propagation of changed IMS segments to DB2 tables. Distinguish between:
- One-way IMS-to-DB2 propagation
- IMS-to-DB2 propagation, as part of two-way propagation

internal segments. Internal Segments is the IMS DPROP and DataRefresher term for structures embedded in IMS Segments, that are propagated through mapping case-3 propagation requests. Each embedded structure (i.e. each internal segment), is propagated to a different table; each occurrence of the embedded structure to one row of the table.

invalid unit of work. An IMS UOW that is missing a first record (containing the INQY data). If the IMS DPROP asynchronous propagation Selector detects an invalid unit, it responds according to what you specified on the INVUOW keyword of the SELECT control statements. If you specified:

IGNORE
: The Selector continues processing

STOP
: The Selector issues an error message and terminates

ISC. Inter-system communications.

ISPF. Interactive system production facility or Interactive structured programming facility.

IXF. Integrated exchange format.

L

LOG-ASYNC. The IMS log-based, asynchronous propagation functions of IMS DPROP.

Once the IMS log records are archived (IMS Online Logs) or de-allocated (IMS Batch Logs) by IMS and then stored in time-stamp sequence, LOG-DPROP reads the IMS logs to find the changed data and then stores the changed data in PRDS data sets. The Receiver component of IMS DPROP reads the PRDSs, transforms the data into the relational format, and applies the changes to the target DB2 tables.

See asynchronous propagation.

logical concatenated key. See IMS logical concatenated key

M

Map Capture exit (MCE) routine. The map capture exit routine provided by DPROP. MCE is used when you provide mapping information through DataRefresher. MCE is called by DataRefresher during mapping and data extract to perform various validation and checking operations. The IMS DPROP MCE should be distinguished from the DataRefresher Map Capture exit, the DataRefresher routine that calls MCE.

mapping case. A definition of how IMS segments are to be mapped to DB2 tables. IMS DPROP distinguishes between mapping case 1, mapping case 2, and user mapping cases.

mapping case 1. One of the generalized mapping cases provided by IMS DPROP. Mapping case 1 maps one single segment type, with the keys of all parents up to the root, to a row in a single DB2 table.

mapping case 2. One of the generalized mapping cases provided by IMS DPROP. Mapping case 2 maps one single segment type, with the keys of all parents up to the root, plus data from one or more immediately subordinate segment types (with a maximum of one occurrence of each segment type per parent), to a row in a single DB2 table.

mapping case 3. One of the generalized mapping cases provided by IMS DPROP. Mapping case 3 supports the propagation of segments containing embedded structures. A typical example of an embedded structure is a repeating group of fields.

- each embedded structure can be propagated to/from a different table. Mapping case 3 propagates each occurrence of an embedded structure, with the key of the IMS segment, and the keys of the physical parent and ancestor, to/from a row of one DB2 table.
- the remaining data of the IMS segment (that is the fields that are not located in a embedded structure) can be propagated to/from another table.

Mapping Verification and Generation (MVG). An IMS DPROP component that validates the mapping information for each propagation request and stores it in the IMS DPROP directory. For a propagation request belonging to a generalized mapping case, MVG generates an SQL update module. MVG is invoked internally by MCE and MVGU.

Mapping Verification and Generation utility (MVGU). An IMS DPROP utility invoked by the DBA. MVGU creates propagation requests when DataRefresher is not used to provide mapping information (i.e., when you put the mapping information directly into the MVG input tables). MVGU also deletes or rebuilds propagation requests in the IMS DPROP directory.

master table. The IMS DPROP directory master table, that is created when IMS DPROP is initialized. It consists of one row, containing system and error information.

MCE. Map Capture exit routine.

MIT. Master Index Table.

MQ-ASYNC. The MQSeries-based, asynchronous propagation functions of IMS DPROP.

An IMS Data Capture Exit routine provided by IMS DPROP obtains the IMS Database changes in real time from IMS and sends the changes via MQSeries messages to an IMS DPROP Apply program. The Apply program reads the MQSeries messages, transforms the data into relational format, and then applies the new data to the target DB2 tables.

MQ-ASYNC supports both near-real time propagation and automated point-in-time propagation.

MQSeries. A family of IBM licensed programs that provide message queuing services.

MQSeries for OS/390. The members of the MQSeries that run on OS/390 systems.

MSDB. Main storage database.

MSC. Multisystem communication.

MVG. Mapping Verification and Generation.

MVG input tables. A group of DB2 tables into which the DBA stores propagation request definitions when DataRefresher is not used to provide mapping information. Once the propagation requests are stored, the DBA invokes MVGU. MVGU invokes MVG, that

validates the propagation request and copies the mapping definitions from the MVG input tables to the IMS DPROP directory.

MVGU. Mapping Verification and Generation utility.

N

Near RealTime. A delay of only a couple of seconds.

O

OLDS. Online Data Set.

One-way DB2-to-IMS propagation. This is the propagation of changed DB2 tables to IMS segments. Distinguish between:
- One-way DB2-to-IMS propagation
- DB2-to-IMS propagation, as part of two-way propagation

One-way IMS-to-DB2 propagation. This is the propagation of changed IMS segments to DB2 tables. Distinguish between:
- One-way IMS-to-DB2 propagation
- IMS-to-DB2 propagation, as part of two-way propagation

P

PCB. Program communication block.

persistent MQSeries message. An MQSeries message that survives a restart of the MQSeries Queue Manager.

physical concatenated key. See IMS physical concatenated key.

Point In Time Propagation. An Asynchronous propagation is said to operate in 'Point In Time' mode, when the data content of the target databases matches the content of the source databases at a previous, clearly identified Point In Time. For example, a Point In Time Propagation can be used to reflect in the content of the target databases the logical end of a business day, or the logical end of business month, or the end of specific Batch jobstream that updated the source databases.

PR. Propagation request.

PR ID. Propagation request identifier.

PRCT. Propagation Request Control Table

PRDS. Propagation Request Data Set

PRDS register file. A data set created by the IMS DPROP asynchronous propagation Selector that contains details of the associated PRDS.

PRDS register table. An IMS DPROP directory table that is created at the Receiver site when IMS DPROP is installed. The table is initially empty and you must populate it, using the PRU REGISTER control statements.

PRDS Registration utility (PRU). An IMS DPROP asynchronous propagation utility that registers PRDSs in the PRDS Register Table.

propagation. See data propagation.

Propagation Data Stream. A stream of changed IMS data that flows in MQSeries messages from the Capture Component of IMS DPROP to the Apply Component of IMS DPROP. Propagation data streams are defined with PRSTREAM control statements in the //EKYTRANS file of EKYMQCAP.

propagation delay. The time elapsed between the update of the IMS source database by the application programs and the update of the target DB2 table by IMS DPROP.

Propagation exit routine. An IMS DPROP exit routine you can write to propagate data when the generalized mapping cases don't meet your needs. A Propagation exit routine must provide all the logic for data mapping, field conversion, and propagation.

propagation group. A subset of the propagation requests in the IMS DPROP directory propagation request table (IMS DPROP asynchronous only).

You can define as many propagation groups as you like, but any propagation request can be associated with one and only one propagation group.

propagation log records. IMS log records that the IMS DPROP asynchronous propagation Selector writes to PRDSs:
- 9904 (update) records
- Commit or abort records
- SETS/ROLS records

propagation request control table (PRCT). An IMS DPROP directory table that is created at the Receiver site when IMS DPROP is installed. It contains details of all propagation requests defined to IMS DPROP and, in combination with the RCT, enables the Receiver to ascertain:
- Which propagation requests are assigned to which Receivers
- The activity status of all defined Receivers
- The activity status of all propagation requests that are assigned to defined Receivers

Propagation Request data set (PRDS). A sequential file into which the IMS DPROP asynchronous propagation Selector writes all propagation log records for a propagation group.

propagation request (PR). A request to propagate data between IMS and DB2. You define propagation requests for each segment type that is to be propagated.

PR set. A group of logically related propagation requests, identified by having the same PRSET ID. PR sets are typically used when you propagate the same IMS data to multiple sets of DB2 tables.

PRU. PRDS Registration utility.

PSB. Program specification block.

R

RCT. Receiver control table.

Receiver. An IMS DPROP asynchronous propagation component that retrieves the propagation log records from a PRDS and passes them to the RUP, that uses them to update the DB2 target tables.

Applies to LOG-DPROP.

RECEIVER control statement. A control statement that is input directly into the IMS DPROP asynchronous propagation Receiver JCL to specify:

- The name of the Receiver that is to process a PRDS
- The names of the DB2 subsystem to be accessed and the DB2 plan
- The number of committed UOWs to process before a DB2 commit is issued

Applies to LOG-DPROP.

Receiver control table (RCT). An IMS DPROP directory table, that is created at the Receiver site when IMS DPROP is installed. The table is initially empty and you must populate it, using the SCU CREATEREC control statement. It contains details of all Receivers and, in combination with the PRCT, enables the Receiver to ascertain:

- Which propagation requests are assigned to which Receivers
- The activity status of all defined Receivers
- The activity status of all propagation requests that are assigned to defined Receivers

Applies to LOG-DPROP.

Relational to Hierarchical propagation. This is one-way relational to hierarchical propagation: the one-way propagation of changed DB2 tables to IMS segments. The terms *relational to hierarchical propagation* and *one-way DB2-to-IMS propagation* are interchangeable.

relational update program (RUP). The IMS DPROP component that does the actual IMS to DB2 propagation. RUP is the IMS DPROP-provided IMS Data Capture exit routine.

For synchronous propagation, the IMS Data Capture function calls RUP with the changed IMS segments.

For user asynchronous propagation, your routine gets the changes from IMS and later calls RUP.

For IMS DPROP asynchronous propagation, the Receiver gets the changes from the Selector-Receiver Interface and later calls RUP. In either case, RUP propagates the changes to DB2.

RIR. RIR is an IMS DPROP abbreviation for DB2 Referential Integrity Relationship. Database administrators can define RIRs between tables in order to request that DB2 catches and prevents update anomalies in the relational databases.

Implementation of RIRs between propagated tables is:

- Optional for one-way IMS to DB2 propagation
- Strongly recommended for DB2 to IMS and two-way propagation

RTT. Resource translation table.

RUP. Relational Update program.

RUP control block table. A single IMS DPROP directory table that contains one RUP propagation control block (PRCB) for each propagated segment type. Each RUP PRCB contains details of the relevant database and segment.

S

SCF. Selector Control File.

SCF Apply job. Uses the SCF control statements to create new propagation groups and to list and modify existing propagation groups in the SCF.

SCF Compare job. Used to compare the contents of the Group Definitions File with the propagation groups in the SCF and to generate SCF control statements to bring the SCF into line with the Group Definitions File.

SCF control statements. Can be generated automatically by the IMS DPROP asynchronous propagation GUU or input directly into the IMS DPROP asynchronous propagation SCF Apply utility JCL. The control statements modify the contents of the SCF records.

SCU. Status Change utility.

segment exit routine. An IMS DPROP exit routine you can write to complement the logic of the generalized mapping cases. Segment exit routines are typically used to convert a changed data segment from the form it has in your IMS database to a form you have defined in your propagation request.

SELECT control statements. Control statements that are input directly into the IMS DPROP asynchronous propagation Selector JCL to define the execution options for the Selector.

Applies to LOG-DPROP.

Selector. An IMS DPROP asynchronous propagation component that collects propagation log records from the IMS log files and writes them to PRDSs for later processing by the IMS DPROP asynchronous propagation Receiver component.

Applies to LOG-DPROP.

Selector control file. Created at Selector installation or generation time and contains the following control information that is essential to the operation of the Selector:

- Database records and propagation group records
- DBRC information
- Timestamp information

Applies to LOG-DPROP.

SLDS. System Log Data Set.

SNAP. system network analysis program

Source System. An OS/390 system where IMS source databases of the IMS DPROP propagation reside.

SQL update module. A module generated by MVG for each propagation request belonging to a generalized mapping case. An SQL update module contains all the SQL statements required to propagate to DB2 the changed IMS data for that propagation request.

SSM. Subsystem member. An IMS JCL parameter that identifies the PDS member that describes connection between IMS and the DB2 subsystems.

Status Change utility (SCU). An IMS DPROP utility that:

1. Changes the status of propagation requests in the synchronous environment. Propagation requests can be active, inactive, or suspended. The SCU also performs a variety of other service functions.
2. Maintains the Timestamp Marker Facility and populates the RCT and the PRCT in IMS DPROP asynchronous propagation.

synchronous propagation. The propagation of data within the same unit-of-work as the update call.

T

Target System. An OS/390 system where DB2 target tables of the IMS DPROP propagation reside.

Timestamp Marker Facility. Supports the statements that create, assign, and delete timestamp markers in the SCF. It is run as part of the SCU.

TSMF. Timestamp Marker Facility.

TSMF Callable Interface. A facility that allows a user application to create a stop timestamp for one or more propagation groups.

Two-way propagation. The combination of IMS-to-DB2 propagation and DB2-to-IMS propagation for the same data.

TW propagation. See two-way propagation.

U

UIM. User Input Manager.

ULR. Uncommitted Log Record.

uncommitted log records (ULR). When the IMS DPROP asynchronous propagation Selector terminates, it writes all uncommitted log records (propagation log records that have not yet been either committed or aborted by IMS) to the uncommitted log record data set. On a subsequent Selector execution, these records will be either written to the appropriate PRDS (if they have been committed by IMS) or deleted from the uncommitted log record data set (if they have been aborted by IMS).

UOW. Unit of work.

USER-ASYNC. The User asynchronous propagation functions of IMS DPROP.

user exit. See exit routines.

User Input Manager (UIM). A DataRefresher component to which you describe your IMS databases and the mapping between IMS databases and DB2 tables. The mapping is defined by submitting extract requests. You can specify on an extract requests that the UIM is to invoke the DataRefresher Map Capture exit routine provided by IMS DPROP and pass it the DataRefresher mapping definitions of the extract request.

user mapping case. A mapping case you can develop if the generalized mapping cases don't meet your needs.

V

Virtual Lookaside Facility (VLF). An MVS/ESA component that is a specific implementation of data spaces. IMS DPROP exploits VLF for a high-performance retrieval of mapping information and other control information.

VLF. Virtual Lookaside Facility.

Bibliography

The IMS DataPropagator for z/OS Version 3 Release 1 Library

Order Number	Book Title
SC18-7048	Administrators Guide for Log Asynchronous Propagation
SC18-7056	Administrators Guide for MQSeries Asynchronous Propagation
SC18-7055	Administrators Guide for Synchronous Propagation
SC18-7047	Concepts
SC18-7054	Customization Guide
GC18-7053	Diagnosis
GC18-7049	An Introduction
GC28-7051	Installation Guide
G18-7050	Messages and Codes
SC18-7052	Reference
GC27-1628	Licensed Program Specification

Other Books Referenced in This Book

The following books are referred to in this book or might be helpful in understanding IMS DPROP control statements and utilities:

SC26-4888 DB2 Administration Guide, Version 3

SC26-3265 DB2 Administration Guide, Version 5

SC26-4891 DB2 Command and Utility Reference, Version 3

SC26-3267 DB2 Command Reference, Version 5

SC26-3395 DB2 Utility Guide and Reference, Version 5

SC26-4890 DB2 SQL Reference, Version 3

SC26-3270 DB2 SQL Reference, Version 5

SC26-4248 DXT Reference

SC26-4636 DXT Writing Exit Routines

SH19-6999 DataRefresher Command Reference

SH19-6998 DataRefresher Exit Routines

SC26-3066 IMS/ESA Application Programming: Design Guide, Version 4

SC26-8016 IMS/ESA Application Programming: Design Guide, Version 6

SC26-3062 IMS/ESA Application Programming: DL/I Calls, Version 4

SC26-8015 IMS/ESA Application Programming: DL/I Calls, Version 6

SC26-3065 IMS/ESA Administration Guide: Database Manager, Version 4

SC26-8012 IMS/ESA Administration Guide: Database Manager, Version 6

SC26-3064 IMS/ESA Customization Guide, Version 4

SC26-8020 IMS/ESA Customization Guide, Version 6

SC26-3076 IMS/ESA Installation Volume 2: System Definition and Tailoring, Version 4

SC26-8024 IMS/ESA Installation Volume 2: System Definition and Tailoring, Version 6

SC26-4329 IMS/ESA Utilities Reference: System, Version 4

SC26-8035 IMS/ESA Utilities Reference: System, Version 6

SC26-3072 IMS/ESA Operations Guide, Version 4

SC26-3072 IMS/ESA Operations Guide, Version 6

SC26-3075 IMS/ESA Administration Guide: System

SC26-4627 IMS/ESA Utilities Reference: Database Manager

GN28-1257 OS/390 MVS Application Development Guide

GN28-1427 OS/390 MVS Initialization and Tuning

GC26-3398 An Introduction to DataPropagator Relational

SC26-3399	IBM DB2 Universal Database™ Replication Guide and Reference
SC28-1873	TSO/E V2R3 Administration
SC26-4557	MVS/DFP Managing Non-VSAM Data Sets
SC26-4562	MVS/DFP V3R3 Access Method Services for the ICF

Index

Special characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Program Number: 5655-E52

Printed in USA

SC27-1210-02