IBM IMS DataPropagator for z/OS

Administrators Guide for Log Asynchronous Propagation

Version 3 Release 1

SC27-1216-02

IBM IMS DataPropagator for z/OS

Administrators Guide for Log Asynchronous Propagation

Version 3 Release 1

SC27-1216-02

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 241.

This edition applies to Version 3 Release 1 of IMS DataPropagator, 5655-E52, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. This edition is available in softcopy format only.

Contents

Figures

About this information

IBM® IMS™ DataPropagator for z/OS® (IMS DPROP) is an IMS tool which allows you to maintain consistency between two copies of the same data, where one copy is stored in an IMS Database Manager (IMS DB) database and the other copy is stored in a DB2 database. IMS DPROP is part of the IBM data replication solution.

This documentation is for people who administer IMS™ DataPropagator™ Version 3 Log Asynchronous. Administration (hereinafter referred to as LOG-ASYNC). It covers the design, implementation, and control of data propagation, and operation in the data propagation environment. The information describes tasks that you, as administrator, perform. The following areas are covered:

- Database administration which consists of:
  - Defining data propagation
  - Extracting and loading data
  - Establishing access privileges and restrictions
  - Keeping data consistent
  - Monitoring data propagation
- System administration which consists of:
  - Executing IMS DPROP components
  - Controlling IMS DPROP
  - Tuning the system once IMS DPROP is installed
- Operations administration which consists of:
  - Monitoring and managing the effect of data propagation on system performance
  - Describing changes you should make to your operational procedures to accommodate IMS DPROP

To use the procedures in this guide, you must have already installed IMS DPROP using the installation process provided in the Program Directory included with the product.

This information is available in softcopy formats and on the z/OS Software Product Collection Kit, SK3T-4270. Always check for the most current version by going to the IBM® Data Management Tools Web site at www.ibm.com/software/data/db2imstools/library.html

Changes for IMS DPROP for z/OS Version 3 Release 1

This edition, which is available in softcopy format only, includes technical and editorial changes. IMS DPROP Version 3, Release 1 presents improvements to both the product and the product library.

Product changes

IMS DataPropagator V3 Release 1 has added near-real time propagation with Message Queue MQSeries® Asynchronous propagation (MQ-ASYNC).

Product library changes

The Version 3.1 library has been improved. The *Installation* manual,*Reference*,*Administrators Guide* and*Messages and Codes* manual have been updated.

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

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

A new book, *IMS DataPropagator for z/OS: Concepts*, has been published which replaces part 1 of the previous Version 2 Release 2 Administration Guide and provides an overall conceptual description of data propagation.

Types of data propagation

This information covers the following types of asynchronous propagation:

LOG Asynchronous propagation
Using IMS DPROP components to propagate changes made to a set of IMS databases to a corresponding set of DB2® databases. IMS changes are logged and are applied to DB2 tables at a later time.

User asynchronous propagation (USER-ASYNC)
USER-ASYNC mode propagates changed data from IMS databases to DB2 tables with custom-written programs. These changes are first logged and then applied to the DB2 tables at a later time.

How this information is organized

This information is divided into five parts:

- " Part 1, "Overview of administrative tasks," on page 1," presents an overview of administrator tasks.
- " Part 2, "Mapping and designing your IMS DPROP system," on page 7," covers the mapping and definition phase of data propagation. It describes decisions you must make and rules and guidelines to follow as you design your LOG-ASYNC environment. Part 2 consists of chapters 2-4.
- " Part 3, "Setting Up for Data Propagation," on page 87," covers the setup phase of data propagation, including extracting and loading data. It describes tasks you need to complete to set up and prepare for implementation of IMS DPROP. Part 3 consists of chapters 5-9.
- " Part 4, "Propagating data with IMS DPROP," on page 145," covers the actual propagation phase and the maintenance and control phase of data propagation. It describes tasks to operate, maintain, and tune IMS DPROP. Part 4 consists of chapters 10-15.
- Part 5, offers additional information about JCL, storage requirements, and other aspects of data propagation.

Part 5 consists of four .

- Appendix A, "JCL information," on page 227, describes IMS DPROP JCL Information.

- Appendix B, “LOG-ASYNC Propagation data sets and storage requirements,” on page 229, describes IMS DPROP storage requirements in IMS regions doing LOG-ASYNC propagation.
- Appendix C, “Converting PRTYPE=F into PRTYPE=E propagation requests,” on page 235, tells IMS DPROP R1 users how to convert IMS DPROP R1 TYPE=F to the more powerful R2 TYPE=E.
- Appendix D, “Example of LOG-ASYNC propagation,” on page 237, describes some commands and control statements that you would use in a simple asynchronous propagation scenario.

Terms

The following terms are synonymous in this documentation:

- *File* and *data set*.
- Databases that have been *quiesced* or set to *read-only status*.

  In all cases, these terms refer to:
  - Any database you can propagate, except for DEDBs, that is set to read-only status
  - DEDBs that were taken offline with a /DBR command
- *Data Extract (DXT™)* and *DataRefresher™*.

  Unless stated otherwise, these terms refer to either of the following products:
  - DXT Version 2 Release 5
  - DataRefresher Version 1 or higher

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

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

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

How to use this information

Key administrative tasks for design, setup, and implementation of data propagation are listed in Chapter 1, “Administrative tasks for LOG-ASYNC ,” on page 3. The order in which tasks are presented is the recommended order, but not required.

This documentation contains information that helps you perform administrative tasks. The information is applicable to LOG Asynchronous propagation (LOG-ASYNC).

To use these topics, you should have a working knowledge of:

- what data propagation is and the business reasons for propagating data. Additional information on these topics can be found in *IMS DataPropagator for z/OS: An Introduction*
- *IMS DataPropagator for z/OS: Concepts*
- IMS, DB2, and DataRefresher concepts and functions

Specific changes since the previous edition of this book are indicated by a vertical bar (|) to the left of a change. Editorial changes that have no technical significance are not noted.

Always check the DB2 and IMS Tools Library page for the most current version of this publication:

www.ibm.com/software/data/db2imstools/library.html

Service updates and support information

To find service updates and support information, including software fix packs, PTFs, Frequently Asked Question (FAQs), technical notes, troubleshooting information, and downloads, refer to the following Web page:

www.ibm.com/software/data/db2imstools/support.html

Receiving information updates automatically

By registering with the IBM My Support service, you can automatically receive a weekly e-mail that notifies you when new DCF documents are released, when existing product documentation is updated, and when new product documentation is available. You can customize the service so that you receive information about only those IBM products that you specify.

To register with the My Support service:

1. Go to http://www.ibm.com/support/mysupport
2. Enter your IBM ID and password, or create one by clicking **register now**.
3. When the My Support page is displayed, click **add products** to select those products that you want to receive information updates about. The DB2 and IMS Tools category is located under **Software** -> **Data and Information Management** -> **Database Tools & Utilities**.
4. Click **Subscribe to email** to specify the types of updates that you would like to receive.
5. Click **Update** to save your profile.

Where to find information

The DB2 and IMS 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 page:

www.ibm.com/software/data/db2imstools/library.html

IBM Redbooks™ that cover DB2 and IMS Tools are available from the following Web page:

www.ibm.com/software/data/db2imstools/support.html

How to look up message explanations

You can use any of the following methods to search for messages and codes:

Searching an information center

In the search box that is located in the top left toolbar of any Eclipse help system, such as the IBM Information Management Software for z/OS Solutions Information Center, enter the number of the message that you want to locate. For example, you can enter DFS1065A in the search field.

Use the following tips to help you improve your message searches:

- You can search for information on codes by entering the code; for example, enter -327.
- Enter the complete or partial message number. You can use wild cards (* or ?) in the message number to broaden your search; for example, DFS20??I.

The information center contains the latest message information for all of the information management products that are included in the information center.

Using a Web search

You can use any of the popular search engines that are available on the Web to search for message explanations. When you type the specific message number or code into the search engine, you will be presented with links to the message information in IBM information centers.

Using LookAt

LookAt is an online facility that you can use to look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, VSE/ESA™, and Clusters for AIX® and Linux®:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OSe systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX® System Services running OMVS).
- Your Microsoft® Windows® workstation. You can install code to access IBM message explanations on the z/OS Collection (SK3T-4269) using LookAt from a Microsoft Windows command prompt (also known as the DOS command line).
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from a disk on your z/OS Collection (SK3T-4269) or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Accessibility features

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use a software product successfully. The major accessibility features in this product enable users to:

- Use assistive technologies such as screen readers and screen magnifier software. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.
- Customize display attributes such as color, contrast, and font size.
- Operate specific or equivalent features by using only the keyboard. Refer to the following publications for information about accessing ISPF interfaces:
  - *z/OS ISPF User's Guide, Volume 1*, SC34-4822
  - *z/OS TSO/E Primer*, SA22-7787
  - *z/OS TSO/E User's Guide*, SA22-7794

  These guides describe how to use ISPF, including the use of keyboard shortcuts or function keys (PF keys), include the default settings for the PF keys, and explain how to modify their functions.

People with limited vision who use screen reader software might find the following feature requires particular attention:

Pop-up windows

This product uses ISPF function that produces pop-up windows for some tasks. The pop-up and its frame are just text that overlays the underlying information on the displayed panel. The frame of such a pop-up is not usually recognized as such by screen reader software, so you may need to gain some familiarity with reading such panels before the information becomes meaningful. Alternatively, you can display the pop-up window on a full screen by using the RESIZE command or function key.

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 information or any other IMS DataPropagator for z/OS documentation, use either of the following options:

- Use the online reader comment form, which is located at:

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

Part 1. Overview of administrative tasks

Chapter 1, “Administrative tasks for LOG-ASYNC ,” on page 3, lists the tasks that you, as IMS DPROP administrator perform to design, setup, and perform propagation.

Chapter 1. Administrative tasks for LOG-ASYNC

This section summarizes tasks that you perform for LOG-ASYNC propagation:
- Map data and design for propagation
- Set up IMS DPROP, IMS, and DB2
- Propagate data
- Maintain and control your propagation environment

Each task listed in Table 1 is followed by a reference that tells you where you can find more information about the task.

You can vary the order in which you complete tasks depending on your needs. The order in which tasks are presented in this table is the recommended order, but not a required order. You might also repeat tasks in various phases of propagation. For example, you extract and load data during the setup phase, but you also can extract and load data during the maintenance and control phase in order to synchronize the data.

Tasks for implementing Log-Asynchronous propagation

Table 1 summarizes tasks you complete to prepare for and implement LOG-ASYNC and user LOG-ASYNC propagation. The tasks are divided into phases:
- Mapping and design: choosing which data to propagate
- Setup: ensuring that your IMS and DB2 systems are ready for propagation
- Propagation: beginning and refining propagation
- Maintenance and control: checking for data consistency or to adjust propagation requests

Table 1. Summary of task steps for LOG-ASYNC propagation

1. Install IMS DPROP. Refer to the *Installation Guide* for details.
Mapping and Design Phase
2. In IMS DPROP, define the mappings. • Identify mapping requirements. • Select the IMS segments and data fields that are to be propagated. • Identify DB2 tables and columns. • Choose the mapping case or design your own mapping. See Part 2, “Mapping and designing your IMS DPROP system,” on page 7.
3. Follow rules and adhere to limitations for IMS data. • Check field specs (for example, numeric fields must be truly numeric for IMS and DB2). • Check variable-length IMS segments to ensure propagated IMS fields are wholly contained in the segment occurrence. See “IMS environment” on page 84. Also, see “Supported field formats and conversions” on page 64 for specifics on field specifications and “Propagating variable-length segments (IMS-to-DB2)” on page 41 for specific rules and limitations.
4. Create exit routines, if needed. Exits must be coded and compiled into the IMS DPROP library. Refer to *Concepts* for a general description of when to use exit routines, and refer to the *Customization Guide* for details on creating exit routines.
Setup Phase

Table 1. Summary of task steps for LOG-ASYNC propagation (continued)

5. In IMS, register each database which is a source of propagation data with DBRC using the INIT.DB command. Registration is required for LOG-ASYNC propagation. See “Use of DBRC” on page 84 for an explanation of this requirement. Refer to *IMS/ESA Utilities Reference: System* for details on the INIT.DB command.
6. In IMS, modify (or create) the database definition (DBD) of each database to specify IMS CDC (9904) records to be logged. See “Creating or changing DBDs” on page 89. In this step, you identify the segments which are subject to propagation and provide MVGIN segment structures. Run DBDGEN after creating or modifying the DBDs. If the database is referred to in an online IMS environment, you must also run ACBGEN.
7. If the propagated DB2 tables do not exist, create them in DB2. The tables must exist before you use the MVG to create propagation requests. For additional information, see “Creating DB2 tables” on page 96.
8. Use the DataRefresher MCE or the IMS DPROP MVGU to: • Populate MVG tables with propagation request definitions. • Build propagation requests. See Chapter 6, “Defining and changing propagation requests,” on page 101.
9. If you are not using the DB2 package bind function, then, in DB2 bind the DB2 plans with the new or changed DBRMs. (If you use the package bind function, this step is not necessary.) You are binding the Receiver plans. For authorization issues, see “Binding DB2 plans of propagating applications” on page 124. For specifics about binding for LOG-ASYNC propagation, see “Binding the Receiver and other plans” on page 97. For information about how to bind, see Chapter 8, “Binding and administering plans,” on page 127.
10. In IMS DPROP, use the SCU to create Receivers and define propagation groups as collections of one or more related propagation requests. In this step, you are also constructing SCF statements. For more information, see “Creating, modifying, and deleting propagation groups” on page 98.
11. In IMS DPROP, update the SCF to contain a list of all IMS subsystem identifiers (SSID) from which propagation can occur. Use the SCF Apply utility and the ADDSSID control statement to define SSID and add the SSID to propagation groups. For a brief explanation, see “Adding SSIDs to propagation groups” on page 99. For details on the utility and control statement, refer to the *Reference* .
12. In IMS, establish a synchronization point from which to start propagation. You are setting time markers. Coordinate this timestamp activity with the extract and load of data. For procedure steps, see “Establishing the start conditions for LOG-ASYNC propagation” on page 100. For synchronization suggestions and methods, see “Setting synchronization points” on page 192.
13. Extract and load data. Use SCU and DataRefresher or your own programs. During this activity, you are making DB2 tables which are copies of IMS databases. For more information, see Chapter 9, “Extracting and loading data for IMS-to-DB2 propagation,” on page 137.
14. Optional: Modify the IMS Archive JCL so that records involved in propagation are written to separate data sets (CDCDS) to improve performance in the Selector. For more information, see “Criteria for deciding whether to use SLDSs or CDCDSs” on page 220 and “Modifying the archive JCL to create CDCDSs” on page 227.
Propagation Phase (Regularly Scheduled Activity)

Table 1. Summary of task steps for LOG-ASYNC propagation (continued)

15. In IMS DPROP, run the Selector to retrieve propagation log records from either IMS (archive) logs or CDCDS (if used) and to write them to PRDS. When running the Selector, use the SELECT control statement to specify propagation groups to be used and the start and stop time for the propagation groups. For more information, see “Using the Selector” on page 154. For more information on the SELECT control statement, refer to the *Reference* . This step creates PRDS.
16. If the IMS and DB2 systems are on different MVS images (without shared DASD) transport the PRDSs to the DB2 MVS image, using TSO transmit/receive commands or similar methods. See “Remote site considerations” on page 189.
17. In IMS DPROP, use the PRDS Registration utility (PRU) to register PRDSs in the PRDS register table. See “Registering PRDSs” on page 166.
18. In IMS DPROP, run the Receiver to retrieve propagation log records from the PRDSs and pass them to the RUP, which uses them to update the DB2 tables. See “Using the Receiver” on page 166.
Maintenance and Control Phase (Periodic Activity)
19. Optional: Use the CCU to compare IMS and DB2 data and check for consistency. See Chapter 14, “Verifying Data Consistency (CCU),” on page 203, to understand how you can use CCU and “CCU considerations for LOG-ASYNC propagation” on page 205. Also, refer to the *Reference* for detailed information about the CCU.
20. Delete, replace, and rebuild propagation requests, as needed. See information on deleting, replacing, rebuilding, and revalidating propagation requests beginning on page 114.
21. Revalidate propagation requests, especially if changes have been made to either the IMS DBDs or DB2 tables. See “Revalidating propagation requests” on page 116. You use MVGU to run a REVALIDATE function. The MVGU is described in detail in the *Reference*.
22. Clean up PRDSs and their table entries. Use the UNREGISTER command in the PRU. Then delete the PRDSs from the MVS catalog. If you are using generation data groups (GDGs), the PRDSs will be cleaned up automatically. Refer to the *Reference* for more information on the PRU. Refer to *OS/390 MVS JCL User’s Guide* for more information on GDGs.
23. Clean up CDCDSs and their PRILOG entries in DBRC. If you are using generation data groups (GDGs), the CDCDSs will be cleaned up automatically. Refer to the *IMS/ESA DBRC Guide and Reference* for more information on maintaining the log files.

Part 2. Mapping and designing your IMS DPROP system

Chapter 2. Decisions affecting mapping and propagation

The process of preparing, mapping, and designing propagation involves:
- Determining propagation direction and propagation request type
- Planning your mapping and determining the data to propagate
- Selecting a mapping case for each propagation
- Mapping data and defining propagation requests

This section guides you through the process by providing information on designing propagation. The topics described are:
- Propagation requests, supported propagation directions, and propagation request types.
- The three generalized mapping cases supported by IMS DPROP.
- The PATH data and WHERE clause options associated with the three generalized mapping cases.

Propagation requests and selecting PRTYPEs

A propagation request defines how a particular segment is to be mapped to or from a table. Propagation requests are defined for each segment type or table that is to be propagated. You define propagation requests with either:
- DataRefresher SUBMIT commands
- The MVG input tables

Refer to the *Reference* for detailed information on defining propagation requests with DataRefresher or using the MVGU.

Propagation requests, which are stored in the IMS DPROP directory tables, specify:
- Propagation is to be one-way IMS-to-DB2.
- The propagation request type
- The mapping case number
- Mapping options

See Chapter 6, “Defining and changing propagation requests,” on page 101 for information on creating propagation requests.

Specifying propagation direction

When defining a propagation request, you specify, on the MAPDIR= propagation parameter, in which direction data is to be propagated. The propagation direction supported by IMS DPROP depends on the propagation request type. For LOG-ASYNC, you can only specify hierarchical-to-relational propagation.

As a general rule, two or more propagation requests should have the same MAPDIR value if they are propagating either:
- A group of logically related IMS databases
- The tables of one DB2 referential integrity structure [1]

An exception to this rule applies if you want to propagate the same segment with TW propagation requests and additional HR propagation requests. Additional HR

1. If you need to propagate certain database segments in one direction and other segments of the same database in another direction, you can define all propagation requests with MAPDIR=TW.

propagation requests propagate the segment to tables other than those for the TW propagation requests. Also, for HR propagation requests, consider the following:

- The requests must be PRTYPE=L.
- If your application updates the relevant data, the updates are propagated by both the HR and TW propagation requests.
- If the HUP applies updates to IMS during synchronous DB2-to-IMS propagation, they are *not* propagated by the HR propagation requests, unless the HR propagation requests are defined in another IMS DPROP system and are doing LOG-ASYNC or user asynchronous propagation with the IMS Asynchronous Data Capture function. Consequently, your SQL updates to the tables propagated by synchronous TW propagation requests are propagated to IMS, but are not propagated to the other tables propagated by the HR propagation requests.

  Therefore, when you do SQL updates to the tables propagated by TW propagation requests, you must apply the equivalent SQL updates to the tables propagated by HR propagation requests to ensure consistency between:
  - The tables propagated by the HR propagation requests
  - The IMS segment and tables propagated by the TW propagation requests

Selecting a propagation request type

When you define a propagation request, you specify the type:

PRTYPE
Description

E Extended function propagation request used with generalized mapping. It supports IMS-to-DB2 propagation.

L Limited function propagation requests, used with generalized mapping. It supports only IMS-to-DB2 propagation (synchronous or asynchronous).

U User propagation request, used with user mapping and Propagation exit routines (synchronous or asynchronous).

F Full function propagation request, used with previous IMS DPROP releases.

Because IMS DPROP Version 2 provides the same CCU support for PRTYPE=L and PRTYPE=F, all descriptions apply to both types. PRTYPE=F is rarely referred to explicitly.

When selecting the propagation request type, determine if your mapping can use generalized mapping logic and PRTYPE=E. IMS DPROP provides full support for PRTYPE=E, supporting all types of propagation. Defining your propagation requests as PRTYPE=E allows you to first implement one-way IMS-to-DB2 propagation and then switch to DB2-to-IMS synchronous propagation.

If you cannot define a propagation request as PRTYPE=E, you must choose between PRTYPE=L and PRTYPE=U.

PRTYPE=L

Uses the generalized mapping logic of IMS DPROP. You do not need to provide Propagation exit routines. You can use the CCU with PRTYPE=L but not for PRTYPE=U.

However, PRTYPE=L does not support DB2-to-IMS synchronous propagation and two-way synchronous propagation. Therefore, do not use PRTYPE=L if you intend eventually to implement DB2-to-IMS synchronous propagation.

PRTYPE=U

Uses Propagation exit routines that you provide. You can use specialized mapping logic in the exit routine and, therefore, have more flexibility than with the generalized mapping logic of IMS DPROP. However, you cannot use the CCU and DLU with PRTYPE=U. You must decide if your Propagation exit routine should support IMS-to-DB2 propagation, DB2-to-IMS synchronous propagation, and two-way synchronous propagation.

Table 2 compares propagation request types. For each propagation request type, the table summarizes both IMS DPROP support and requirements. To follow the references to notes in the table, go to the appropriate PRTYPE descriptions in the sections following the table.

Table 2. Characteristics of propagation request types

	PRTYPE=E	**PRTYPE=L**	**PRTYPE=U**
Support Provided by IMS DPROP			
Generalized mapping logic.	Yes (see PRTYPE=E note 1)	Yes (see PRTYPE=L note 1)	No (see PRTYPE=U note 1)
One-way IMS-to-DB2 propagation.	Yes (see PRTYPE=E note 2)	Yes (see PRTYPE=L note 2)	User dependent (see PRTYPE=U note 2)
One-way DB2-to-IMS and two-way synchronous propagation.	Yes (see PRTYPE=E note 2)	No	User dependent (see PRTYPE=U note 2)
Compatible DataRefresher mapping support for the extract.	Yes (see PRTYPE=E note 3)	Yes (see PRTYPE=L note 3)	User dependent (see PRTYPE=U note 3)
Supports DL/I Load utility.	Yes (see PRTYPE=E note 4)	No (see PRTYPE=L note 4)	No (see PRTYPE=U note 4)
Supports CCU.	Yes (see PRTYPE=E note 5)	Yes (see PRTYPE=L note 5)	No (see PRTYPE=U note 5)
CCU can run concurrently to updates.	Full support	Limited support (see PRTYPE=L note 5)	N/A
CCU can create DB2 repair statements.	Yes (see PRTYPE=E note 5)	Yes (see PRTYPE=L note 5)	N/A
CCU can create DL/I repair statements.	Yes (see PRTYPE=E note 5)	No (see PRTYPE=L note 5)	N/A
Compatibility of referential integrity rules checked by IMS DPROP.	Yes (see PRTYPE=E note 7)	Yes (see PRTYPE=L note 7)	No (see PRTYPE=U note 7)
Requirements of IMS DPROP			
Mapping must comply with the requirements of generalized mapping logic.	Yes	Yes	N/A

Table 2. Characteristics of propagation request types (continued)

	PRTYPE=E	**PRTYPE=L**	**PRTYPE=U**
IMS DPROP requirements for key rules.	Strong (see PRTYPE=E note 6)	Weaker (see PRTYPE=L note 6)	No (see PRTYPE=U note 6)
Mapping PATH data requires that PATH=ID be specified *and* that PATH data not change its value.	Yes (see PRTYPE=E note 1)	-	N/A
Segment and Field exit routines must support mapping for DB2-to-IMS synchronous propagation.	Yes (see PRTYPE=E note 8)	No (see PRTYPE=L note 8)	N/A (see PRTYPE=U note 8)
For one-way DB2-to-IMS and two-way synchronous propagation IMS DPROP requires that each parent/ancestor also be propagated (in the same direction) with a PRTYPE=E or PRTYPE=U.	Yes (see PRTYPE=E note 2)	N/A	Yes
Propagation requests that propagate IMS segments must be defined in IMS top-down sequence.	Required or recommended (see PRTYPE=E note 2)	Recommended (see PRTYPE=L note 2)	Required or recommended (see PRTYPE=U note 2)
With few exceptions, an IMS segment can be propagated by only one PRTYPE=E.	Yes (see PRTYPE=E note 9)	N/A (see PRTYPE=L note 9)	N/A (see PRTYPE=U note 9)
Extension segments of a mapping case 2 propagation request cannot have an IMS key field. Dependents of an extension segment cannot be propagated by a PRTYPE=E.	Yes (see PRTYPE=E note 10)	No (see PRTYPE=E note 10)	N/A
Additional requirements.	Yes (See PRTYPE=E notes 11, 12, 13, 14, and 15)	No	No

PRTYPE=E (Extended Function)

This topic gives more detail on the characteristics and requirements of PRTYPE=E, summarized in Table 2 on page 13.

PRTYPE=E has the following characteristics:

1. Mapping, conversions, and propagation are done using IMS DPROP mapping cases 1, 2, and 3.

   The WHERE clause option is supported.

   The PATH data option is supported. PATH data must not change its value and you must define your propagation request with PATH=ID. Refer to “PATH data” on page 27 for more details on this subject.
2. IMS-to-DB2 propagation, DB2-to-IMS synchronous propagation, and two-way synchronous propagation are all supported.
   - For DB2-to-IMS and two-way synchronous propagation, IMS DPROP requires that each physical and logical parent or ancestor of a propagated child segment be propagated with a PRTYPE=E or U and perform all DB2-to-IMS or two-way synchronous propagation.

- For DB2-to-IMS and two-way synchronous propagation, you must define propagation requests that propagate IMS segments in IMS top-down sequence. Define propagation request that propagate a physical and logical parent segment before defining propagation requests for the child segments.
- For IMS-to-DB2 propagation when IMS/DB2 RIRs are implemented between propagated tables, we recommend that you define propagation requests in IMS top-down sequence to reduce the number of IMS DPROP messages indicating that DB2 RIRs do not match IMS physical and logical parent/child relationships.

3. To do an IMS extract and DB2 load, you can use DataRefresher and the DB2 Load utility. IMS DPROP mapping done during propagation is compatible with the mapping done by DataRefresher and the DB2 Load utility during the IMS extract and DB2 load of data.
4. The propagation request is supported by the CCU. For PRTYPE=E, IMS DPROP provides full support for running the CCU concurrently to database updates.

   For PRTYPE=E, the CCU creates both DL/I and DB2 repair statements.
5. A strict set of rules for the mapping of keys exists. See “Key mapping rules by propagation request type” on page 49.
6. IMS DPROP checks that DB2 RIRs are compatible with IMS parent/child relationships. See “DB2 referential integrity guidelines” on page 45 for further discussion.

   For one-way IMS-to-DB2 propagation, DB2 RIRs are optional. For one-way DB2-to-IMS and two-way synchronous propagation, you should implement matching DB2 RIRs.
7. Segment and Field exit routines used with PRTYPE=E must support mapping for both one-way IMS-to-DB2 propagation and DB2-to-IMS synchronous propagation, even if the propagation request is defined for one-way IMS-to-DB2 propagation. This is because exit routines can be called for one-way DB2-to-IMS mapping during CCU and DLU processing.
8. You can propagate the same segment to or from multiple tables with PRTYPE=E only when:
   - IMS segments containing embedded structures are propagated by mapping case 3 propagation requests.
   - PRTYPE=E is defined with a WHERE clause.
9. Extension segments of a mapping case 2 PRTYPE=E cannot have an IMS key field.

   Dependents of extension segments cannot be propagated with PRTYPE=E.
10. An IMS field (or part of one) mapped to the DB2 primary key cannot be mapped to more than one column.
11. You must provide a Segment exit routine for a segment propagated by mapping case 3.

    You must define internal segments (also called embedded structures) propagated by mapping case 3 as having a variable number of occurrences. You cannot propagate the counter field.
12. All IMS fields in an entity segment that are included in a WHERE clause must be mapped to the DB2 table.
13. For an IMS unidirectional logical relationship, the IMS delete rule for the logical parent segment must be PHYSICAL. For PRTYPE=L and U, the delete rule can be either PHYSICAL or LOGICAL.

14. If you are implementing a DB2 RIR matching an IMS logical parent/child relationship, an IMS PHYSICAL delete rule for the logical parent should be matched with a DB2 delete rule of ON DELETE CASCADE. For PRTYPE=L, the DB2 delete rule can be ON DELETE RESTRICT or ON DELETE CASCADE for PRTYPE=U. No rules are imposed for DB2 RIRs.

PRTYPE=L (Limited Function)

This topic gives more detail on the characteristics and requirements of PRTYPE=L, summarized in Table 2 on page 13.

PRTYPE=L has the following characteristics:

1. Mapping, conversions, and propagation are done using mapping cases 1, 2, and 3.

   The WHERE clause option is supported.

   The PATH data option is supported. PATH data is supported both for the PATH=ID option (which requires that PATH data not change its value) and for the PATH=DENORM option (which allows PATH data to change its value). See “PATH data” on page 27 for more details on this subject.
2. Only one-way IMS-to-DB2 propagation is supported.

   If you implement IMS/DB2 RIRs between propagated tables, we recommend that you define propagation requests in hierarchical IMS top-down sequence to reduce the number of IMS DPROP messages indicating that DB2 RIRs do not match IMS physical and logical parent/child relationships.
3. To do an IMS extract and DB2 load, you can use DataRefresher and the DB2 Load utility. IMS DPROP mapping done during propagation is compatible with the mapping done by DataRefresher and the DB2 Load utility during the IMS extract and DB2 load of data.
4. The propagation request is not supported by the DLU.
5. The propagation request is supported by the CCU.

   IMS DPROP provides limited support for running the CCU concurrently with database updates. The CCU might inadvertently identify some DB2 rows as unmatched with an IMS segment occurrence.

   The CCU creates DB2 repair statements but no DL/I repair statements.
6. A set of rules for the mapping of keys exists, but they are less restrictive than those for PRTYPE=E. See “Key mapping rules by propagation request type” on page 49.
7. IMS DPROP checks that DB2 RIRs are compatible with IMS parent/child relationships. See “RIR matching a physical IMS parent/child relationship” on page 47 for a complete discussion.

   DB2 RIRs are optional.
8. Segment and Field exit routines used with PRTYPE=L support only IMS-to-DB2 mapping.
9. You can propagate the same segment to multiple tables with PRTYPE=L.
10. Extension segments of a mapping case 2 PRTYPE=L propagation request can have an IMS key field.

    Dependents of extension segments can be propagated with PRTYPE=L propagations requests.

PRTYPE=U (User Mapping)

This section gives more detail on the characteristics and requirements of PRTYPE=U, summarized in Table 2 on page 13.

Use PRTYPE=U for propagation requests when you do not use the generalized mapping cases. PRTYPE=U has the following characteristics:

1. A Propagation exit, that you write, maps, converts, and propagates data.
2. Your Propagation exit routine provides support for IMS-to-DB2 propagation, DB2-to-IMS synchronous propagation, and two-way synchronous propagation.
   - For DB2-to-IMS and two-way synchronous propagation, IMS DPROP requires that each physical and logical parent or ancestor of a propagated child segment be propagated with a PRTYPE=E or U and perform all DB2-to-IMS or two-way synchronous propagation.
   - For DB2-to-IMS and two-way synchronous propagation, you must define propagation requests that propagate IMS segments in IMS top-down sequence. Define propagation requests that propagate a physical and logical parent segment before defining propagation requests for the child segments.
   - For IMS-to-DB2 propagation when IMS/DB2 RIRs are implemented between propagated tables, we recommend that you define propagation requests in IMS top-down sequence to reduce the number of IMS DPROP messages indicating that DB2 RIRs do not match IMS physical and logical parent/child relationships.
3. DataRefresher supports IMS extract and DB2 load only if you provide DataRefresher mapping definitions that are compatible with the mapping done by the Propagation exit routine.
4. The propagation request is not supported by the DLU.
5. The propagation request is not supported by the CCU.
6. No rules are imposed or checked by IMS DPROP for the mapping of keys.
7. No rules are checked by IMS DPROP for RIRs defined in DB2.
8. Segment and Field exit routines are not called by IMS DPROP (but can be called by your Propagation exit routine).
9. You can propagate the same segment to or from multiple tables.

As with other propagation request types, IMS DPROP provides the following support for PRTYPE=U:

- Debugging using trace and other facilities
- Centralized error handling
- Dynamic activation, deactivation, and suspension of propagation
- Protection against unintentional updates during DataRefresher extract and load activities
- Centralized control for propagation request definitions (called IMS DPROP directory tables)
- A common process to manage the data propagation environment for both user and generalized mapping

PRTYPE=F (Full Function)

Avoid creating new PRTYPE=Fs. PRTYPE=F is supported only for compatibility with previous IMS DPROP releases. IMS DPROP Version 1 Release 2 and following releases handle PRTYPE=F and PRTYPE=L in the same way.

In Release 1 Version 1, you could define PRTYPE=F, L, and U. The CCU supported PRTYPE=F but not PRTYPE=L.

In Release 1 Version 2 and Release 2 Version 1, CCU supports both PRTYPE=L and PRTYPE=F, with no distinction. For compatibility with Release 1 Version 1, you

can still define PRTYPE=F in Version 1 Release 2 and Version 2 Release 1. Support for PRTYPE=F is the same as for PRTYPE=L. The same rules apply to both propagation request types and the rules are less restrictive than Version 1 Release 1 rules.

If you have PRTYPE=F from a previous release, we recommend that you:

- Install Version 2 Release 1, first. Do not change your existing PRTYPE=F until you are sure migration to Version 2 Release 1 is successful.
- If you need any of the functions provided by the more powerful PRTYPE=E, (for example, DB2-to-IMS synchronous propagation), convert your PRTYPE=F to PRTYPE=E because the R2 rules for PRTYPE=E are stricter than the R1 rules for PRTYPE=F. You might also need to modify your IMS database and DB2 table definitions. See Appendix C, “Converting PRTYPE=F into PRTYPE=E propagation requests,” on page 235.
- If you do not need any of the PRTYPE=E functions, you can either:
  - Leave PRTYPE=F unchanged.
  - Change the PRTYPE=F to PRTYPE=L to avoid confusion. Make this minor change by changing the PRTYPE propagation parameter and recreating the propagation request.

Because IMS DPROP handles PRTYPE=L and PRTYPE=F the same way, mention of PRTYPE=L in this documentation implies both propagation request types.

Mapping case characteristics and rules

IMS DPROP generalized mapping supports mapping cases 1, 2, and 3. You can combine generalized mapping cases with the PATH data option and the WHERE clause option. In addition, you can extend generalized mapping using Segment and Field exit routines that you write. You can also write Propagation exit routines to handle mapping and propagation requirements that do not conform to generalized mapping. Refer to the *Customization Guide* for more information on exit routines.

Mapping case 1

Mapping case 1 propagates one single segment type occurrence to or from a row in a single DB2 table. The segment type being propagated is called the *entity segment*, and represents the same entity that the resulting DB2 row represents. The fields mapped can be:

- IMS keys (or subfields of keys) from the entity segment, its physical parent, and its physical ancestors, up to the root
- Non-key fields from the entity segment

In Figure 1 on page 19, mapping case 1 maps one single segment type occurrence with the keys of the parent and all ancestors up to the root. An occurrence of IMS segment SEGC is mapped to a row of the DB2 table TABC. In the table, KEYC, F4, and F5 are the key and non-key fields from SEGC. KEYA is the key of parent SEGA.

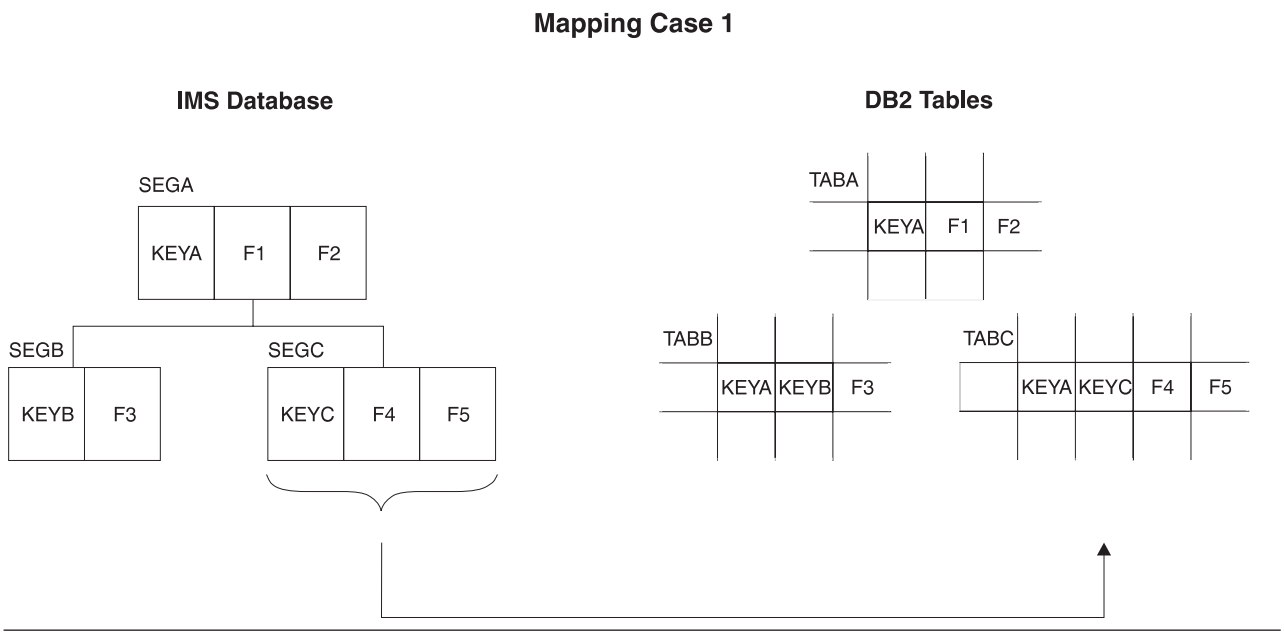


Figure 1. Mapping case 1

Optional:

- You can include non-key fields from each segment in the physical hierarchical path (PATH data option) in the mapping.
- You can make the IMS-to-DB2 propagation dependent on the value of some IMS fields with a WHERE clause. By defining multiple propagation requests with different WHERE clauses for the same segment, you can propagate the same segment to different tables based on field values.

See “Mapping options: Generalized mapping cases only” on page 26 for a detailed description of these options.

Mapping case 2

Mapping case 2 propagates one single segment type occurrence (called the *entity segment*) plus data from one or more immediately subordinate segment types to or from a row in a single DB2 table. Each subordinate segment type included in the mapping can have no more than one occurrence per parent. This type of subordinate segment is called an *extension segment*, and only extends the data in the entity segment. Columns mapped from fields in extension segments must permit a null value or specify NOT NULL WITH DEFAULT. You can map fields that are:

- IMS keys (or subfields of keys) from each segment in the physical hierarchic path and from the entity segment up to the root
- Non-key fields from the entity segment
- From one or more extension segments

In Figure 2 on page 20, mapping case 2 maps one single segment type occurrence with the keys of the parent and all ancestors up to the root. the mapping case also maps data from one or more immediately subordinate segment types (with a maximum of one occurrence of each segment type per parent).

In Figure 2 on page 20, an occurrence of IMS segment SEGC and SEGD are mapped to a row of the DB2 table TABC/D. In the table, KEYA is the key of parent

SEGA. KEYC, F4, and F5 are the key and non-key fields from SEGC. F6 and F7 are non-key fields from the immediately subordinate segment SEGD, which occurs only once.

If propagation is from DB2 to IMS, the arrows in Figure 2 would simply be reversed.

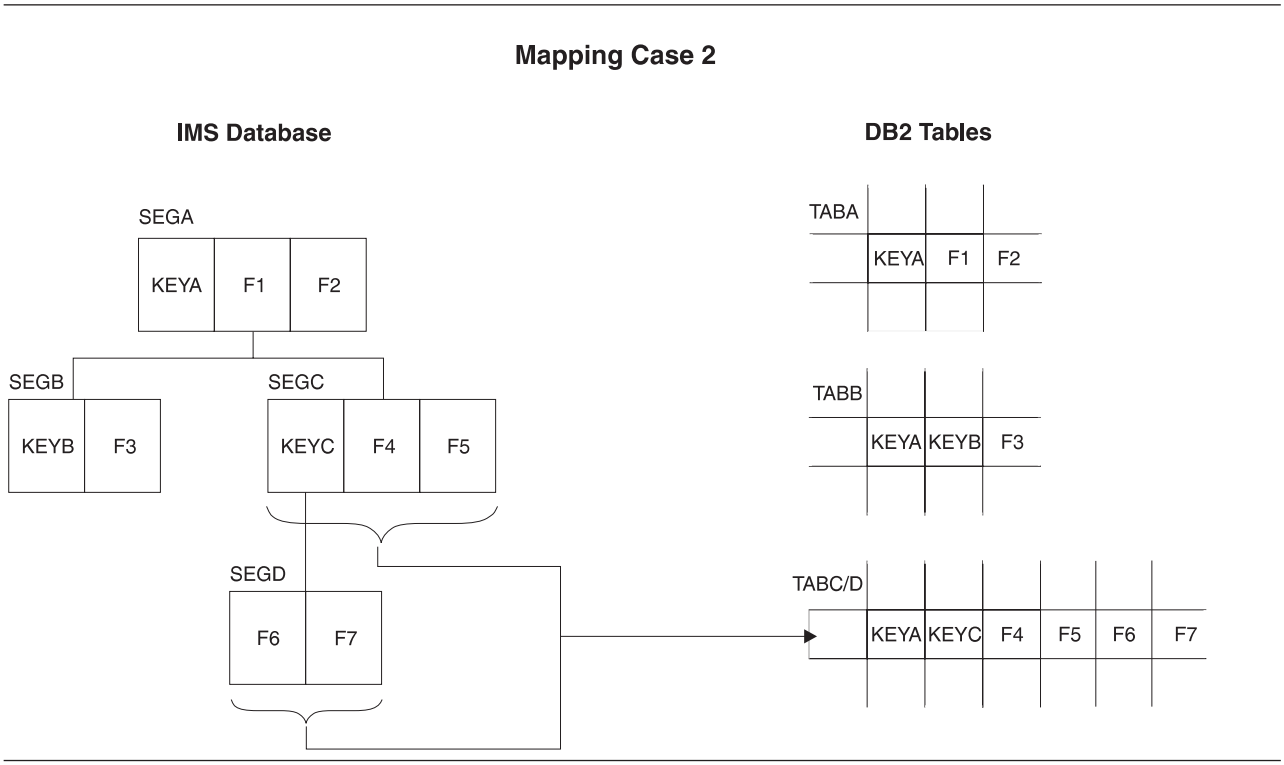


Figure 2. Mapping case 2

Optional:

- You can include non-key fields from each segment in the physical hierarchical path in the mapping (PATH data option).
- You can make the IMS-to-DB2 propagation dependent on the value of some IMS fields with a WHERE clause. By defining multiple propagation requests with different WHERE clauses for the same segments, you can propagate your segments to different tables, based on field values.

See “Mapping options: Generalized mapping cases only” on page 26 for a detailed description of these options.

For HDAM and HIDAM databases, enforce a single occurrence of extension segments under a parent with the POINTER=NOTWIN option in the DBD defining the physical database.

Rules for mapping fields in extension segments

When mapping fields in extension segments, observe the following rules:

1. Fields in an extension segment cannot be mapped to the DB2 primary key.
2. Fields in an extension segment should be mapped to columns that either permit a null value or are defined as NOT NULL WITH DEFAULT.

During IMS-to-DB2 propagation, these columns are set either to a null value or their default value if the extension segment does not exist.

Mapping case 3

Mapping case 3 propagates embedded structures contained in an IMS segment. An *embedded structure* is a group of fields. A typical example of an embedded structure is a repeating group of fields.

An IMS segment can contain one or more embedded structures. Each embedded structure can be propagated by a different mapping case 3 propagation request to or from a different table. A mapping case 3 propagation request maps each occurrence of one embedded structure to or from a row in the DB2 table.

The fields that can be mapped by a mapping case 3 propagation request are:

- IMS keys (or subfields of keys) from each segment in the physical hierarchical path, from the segment containing the embedded structure up to the root
- Fields located in the embedded structure

The fields not located in the embedded structures can be propagated by another propagation request to or from another table. This other propagation request must belong to a mapping case other than mapping case 3 and must conform to the rules for its own mapping case.

In Figure 3 on page 22, mapping case 3 maps embedded structures. Each occurrence of an embedded structure is propagated together with the keys of the physical parent and ancestors up to the root. In Figure 3 on page 22, IMS segment SEGB contains two embedded structures, C and D. C and D occur multiple times within SEGB.

Figure 3 on page 22 shows three different propagation requests.

- PRC is for mapping case 3. Each occurrence of embedded structure C is propagated to one row of TABC, together with the key of segment SEGB and the keys of the physical parent and ancestors up to the root.
- PRD is also for mapping case 3. Propagation is similar except that embedded structure D is propagated to a different DB2 table, TABD.
- PRB is for mapping case 1. The portion of IMS SEGB that did not belong to an embedded structure (together with key fields from each segment in the hierarchic path) is propagated to one row of TABB.

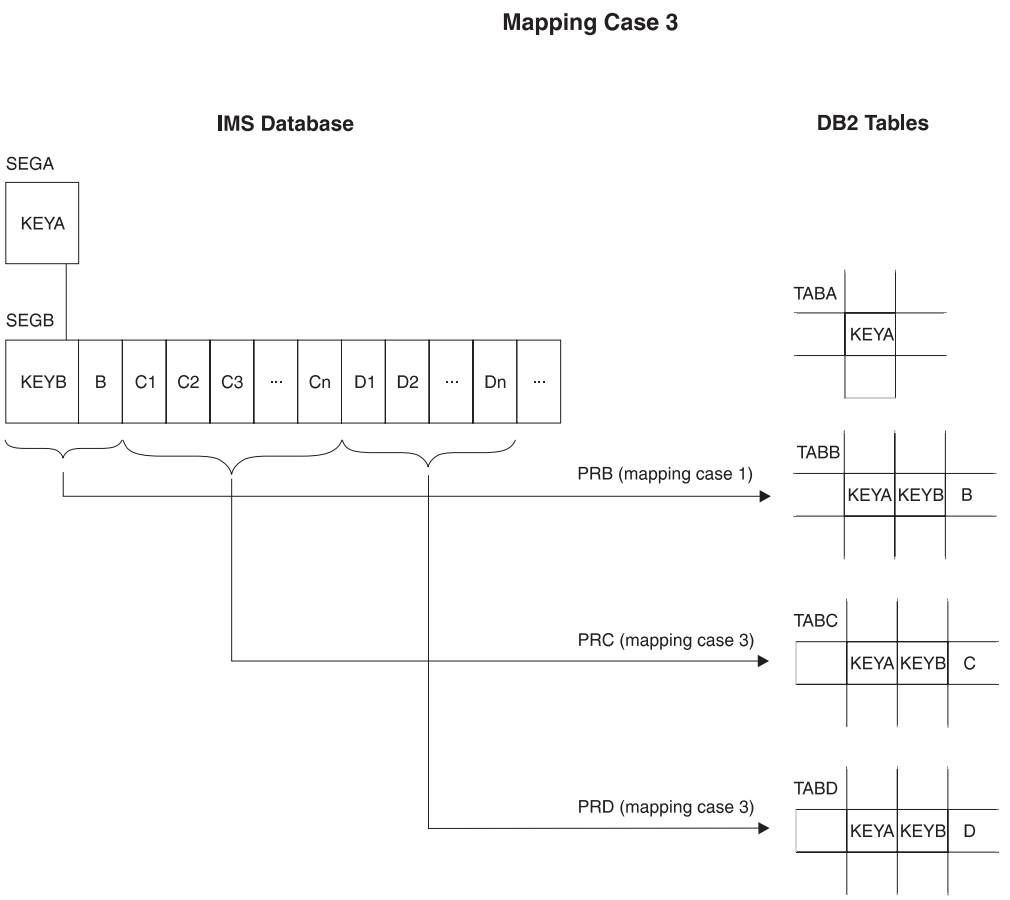


Figure 3. Mapping case 3

For a mapping case 3 propagation request, you can use PATH data to include non-key fields from the IMS segment containing the embedded structure and from each segment in its physical path up to the root. For a detailed description of PATH data, see “Mapping options: Generalized mapping cases only” on page 26.

You cannot combine use of mapping case 3 and the WHERE clause.

The following sections discuss mapping case 3 in association with:
- Definition of containing and internal segments
- Mapping design
- Internal segments and Segment exit routines
- Unique identification of internal segments
- Fixed and variable number of occurrences of internal segments
- PRTYPE=E and internal segments
- DB2-to-IMS propagation of internal segments
- DLU processing of internal segments
- SEG= keyword on IMS DPROP control statements

Definition of containing and internal segments

The definition of containing and internal segments is affected by four basic concepts specific to mapping case 3:

- Propagation involves embedded structures, a concept not defined in IMS. In IMS DPROP and DataRefresher, each embedded structure is called an *internal segment type*. See Figure 4 on page 23.

- The IMS segment itself is called the *containing IMS segment* in IMS DPROP and DataRefresher.
- IMS DPROP views the containing IMS segment as the parent of the internal segments.
- IMS DPROP considers the internal segment to be the entity segment.

Containing Segment and Internal Segment Type

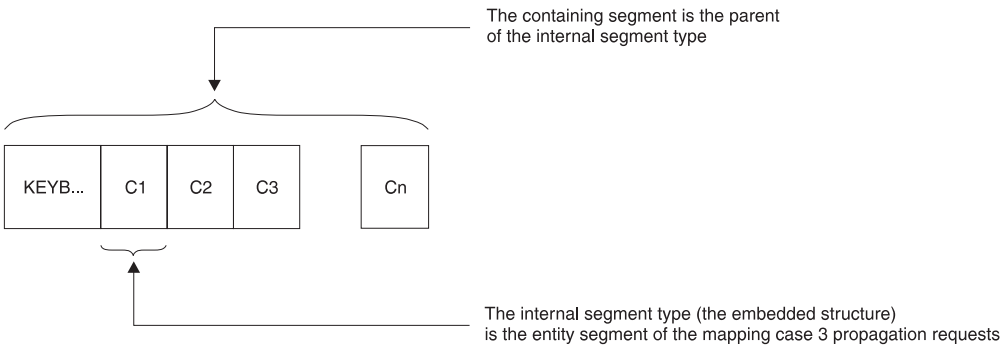


Figure 4. Containing segment and internal segment type

IMS DPROP supports both fixed- and variable-length internal segments. The *Reference* describes how you specify the fixed or variable length of internal segments to IMS DPROP.

IMS DPROP generalized mapping does not support nesting of internal segments. One internal segment cannot contain other internal segments.

When defining multiple propagation requests to propagate containing and internal segments, provide consistent definitions of these segments:

- Define the containing and internal segments in exactly the same way for all propagation requests in the same set. For example, use the same fixed/variable formats, lengths, fixed/variable start positions, and segment names. For information on propagation request sets, see “Using propagation request sets” on page 43.
- Use different names for different internal segment types.

Mapping design for mapping case 3

We recommend that you conceptually transform the IMS database structure into a normalized hierarchical structure before doing your mapping design. This helps you implement IMS DPROP rules as they apply to mapping case 3.

The mapping is a two-step process, as illustrated in Figure 5 on page 24.

1. Decide how you want to transform the IMS database hierarchy (with segments having embedded structures) into a normalized hierarchical structure (with containing and internal segments).
2. Then do your mapping design based on the conceptually normalized structure.

Some IMS DPROP mapping rules (such as matching DB2 RIRs, key mapping, and PATH data) when applied to mapping case 3, apply to mapping between the

normalized hierarchical structure and the *DB2 tables*.

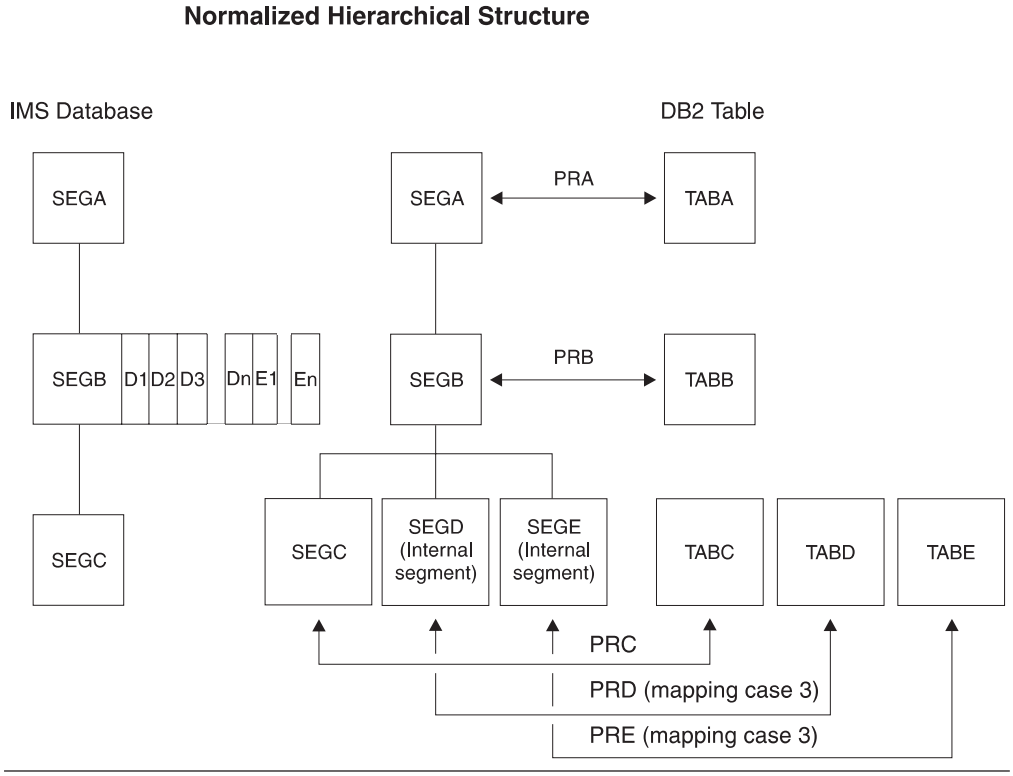


Figure 5. Conceptually normalizing the database for mapping case 3

Figure 5 helps illustrate two steps:

Step 1

Conceptually transform the IMS database structure into a normalized hierarchical structure.

The IMS database structure is shown on the left side of the figure. IMS segment SEGB contains two embedded structures: D and E. In Figure 5, embedded structures D and E occur multiple times within SEGB.

The IMS structure is mapped into a normalized hierarchical structure, as shown in the middle of the figure. IMS segment SEGB is mapped to containing segment SEGB and to internal segments SEGD and SEGE. SEGD and SEGE are children of containing segment SEGB.

Step 2

Using propagation requests, map the normalized hierarchical structure into DB2. Internal segments SEGD and SEGE are mapped with mapping case 3. Other segments are mapped by propagation requests belonging to a different mapping case.

Internal segments and segment exit routines

IMS DPROP and DataRefresher support segment exit routines for the entire IMS segment, not just the internal segment.

You might need to write a segment exit routine for mapping case 3, if you want to:

- Artificially construct, in the internal segment, ID fields uniquely identifying each occurrence of the internal segment.

- Artificially construct, in the containing segment, a counter field to count the number of occurrences of the internal segment type within the containing segment (for internal segments whose number of occurrences varies).
- Support DB2-to-IMS mapping of PRTYPE=E. For PRTYPE=E doing synchronous propagation of internal segments, you must provide segment exit routines.

Because you might need to provide a segment exit routine, you might consider writing a Propagation exit routine and doing your own user mapping. Keep in mind that generalized mapping has advantages over user mapping, such as:

- Segment exit routines are easier to write than Propagation exit routines. You do not need to provide SQL logic.
- You can use the IMS DPROP trace facility for SQL and DL/I updates performed by your propagation request.

Unique identification of internal segments

The IMS DPROP key mapping rules for generalized mapping require that you uniquely identify internal segments with multiple occurrences. Therefore, internal segments with multiple occurrences must have fields that:

- Uniquely identify each occurrence within its containing IMS segment
- Map to the DB2 primary key

Refer to “Key mapping rules by propagation request type” on page 49 for a detailed description of these rules.

However, few internal segments are unique based on field values. You should also consider using a Segment exit routine if you need to propagate internal segments that are not uniquely identifiable. The Segment exit routine can construct ID fields in the edited format of each internal segment occurrence. The value in the ID fields should uniquely identify each occurrence of the internal segment and should be mapped to the DB2 primary key.

For example, if an internal segment was a single field containing yearly revenue with no field for the year, the Segment exit routine could edit the segment format so that each internal segment contained both the year as the ID field and the yearly revenue.

When IMS DPROP updates the containing IMS segment, RUP compares the before and after image of the containing segment. By comparing the ID fields of internal segments, RUP determines which occurrences of internal segments have been inserted, replaced, and deleted. RUP then triggers the appropriate SQL inserts, updates, and deletes for the target DB2 rows.

Fixed/variable number of occurrences of internal segments

IMS DPROP and DataRefresher support both a fixed and variable number of occurrences of the internal segment within the containing segment.

If the internal segment has a *fixed* number of occurrences, you specify this number when defining the propagated IMS segment to IMS DPROP or DataRefresher.

If the internal segment has a *variable* number of occurrences, then the actual number of occurrences must be stored in a count field in the IMS segment. The count field must be located before the first occurrence of the internal segment. If you are defining PRTYPE=E, do not propagate the count field.

If you need to propagate internal segments whose number of occurrences varies and the IMS segment does not contain a count field, consider using a segment exit routine. The segment exit routine can edit the segment and include a count field.

IMS DPROP supports both a variable and fixed number of internal segments for PRTYPE=L. Internal segments must be defined with a variable number of occurrences and with a counter field for PRTYPE=E and synchronous one-way DB2-to-IMS mapping.

If you need to create PRTYPE=E for internal segments that have a fixed number of occurrences, define the number of occurrences as variable. A segment exit routine is required for PRTYPE=E propagating IMS segments containing internal segments and constructs a count field in the edited format of the segment. PRTYPE=E does not map the count field to a column in the DB2 table.

PRTYPE=E and internal segments

You must provide a segment exit routine for IMS segments propagated by propagation requests that are both mapping case 3 and PRTYPE=E.

When a target row of an internal segment is inserted, IMS DPROP does not know the sequence in which your application expects to store the internal segment occurrences within the IMS segment. Your segment exit routine must construct the IMS segment according to the requirements of your IMS applications.

For PRTYPE=E, your segment exit routine is called during mapping for both IMS-to-DB2 propagation and DB2-to-IMS synchronous propagation.

IMS-to-DB2
: Your segment exit routine must map the segment from its IMS format to the format that has been defined to IMS DPROP.

Refer to the *Customization Guide* for detailed information on the logic your exit routine must provide for mapping case 3.

SEG= keyword on IMS DPROP control statements

Many IMS DPROP control statements have a SEG= keyword. When using the SEG= keyword, specify the names of IMS segments, but not the names of internal segments.

For example, if you specify DEACTIVATE SEG=SEGB, IMS DPROP deactivates all propagation requests propagating IMS segment SEGB. This includes propagation requests propagating internal segments within SEGB.

User mapping cases

When you cannot use mapping cases 1, 2, or 3, IMS DPROP allows you to customize mapping by writing Propagation exit routines. Propagation exit routines provide all necessary mapping logic and the propagating SQL calls. Refer to the *Customization Guide* for a complete discussion of Propagation exit routines.

Mapping options: Generalized mapping cases only

You can use two mapping options with generalized mapping cases:

- PATH data, used with mapping cases 1, 2, and 3
- WHERE clause, used with mapping cases 1 and 2

PATH data

PATH data is data from any non-key field in the physical hierarchical path of the entity segment, from the physical parent up to the root. You can include PATH data in propagation requests for mapping cases 1, 2, and 3.

PATH data can come from one or more segments in the physical hierarchic path of the entity segment. Mapping PATH data allows you to combine non-key data from multiple segments in a hierarchical path into one target DB2 table. If the entity segment is an internal segment, PATH data includes non-key fields in the hierarchical path from the containing IMS segment up to the root.

This section discusses:
- Uses of PATH data
- Denormalizing data to improve performance of DB2 queries
- Mapping ID fields of a physical parent/ancestor

Figure 6 on page 28 shows propagation of PATH data. In the figure, PR3, which propagates entity segment SEG3 to TAB3, includes the following in its mapping:
- Key fields from each segment in the physical hierarchical path, from the root down to entity segment SEG3. These fields are KEY1, KEY2, and KEY3.
- PATH data fields from segments in the physical hierarchical path, from the root down to physical parent SEG2. These fields are DATA1 and DATA2.
- Non-key fields from the entity segment. These fields are DATA3 and other fields not shown in Figure 6 on page 28.

By definition, PATH data is always located in a higher hierarchical level than the entity segment, never in a lower level. Also, PATH data is always located in a physical parent/ancestor, not in a logical parent/ancestor.

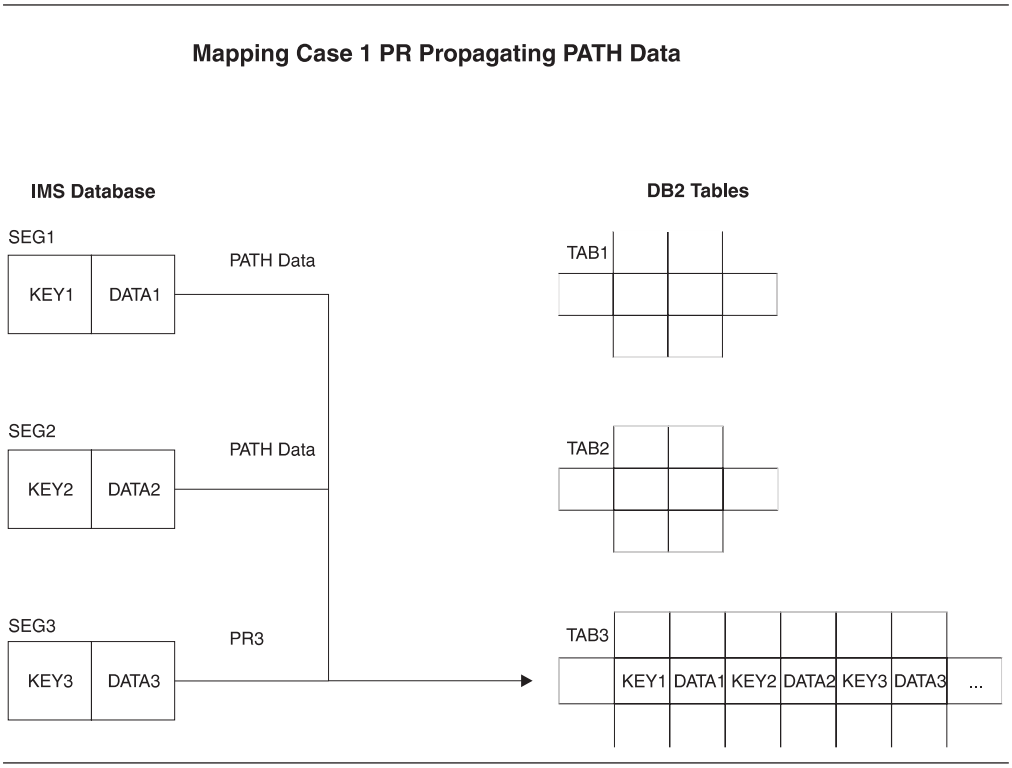


Figure 6. Mapping case 1 propagation request propagating PATH data

Uses of PATH data

IMS DPROP supports two different uses of PATH data. Because the rules for these two uses are different, you must specify which one you are using during propagation request definition on the PATH= parameter of the propagation request:

- PATH=DENORM includes data fields from the physical parent/ancestor that *can change their values*. These fields are often included to intentionally denormalize data to improve the performance of DB2 queries.

  PATH=DENORM applies only to PRTYPE=L and one-way IMS-to-DB2 propagation when the DB2 copy of your data is used for read-only.
- PATH=ID specifies only IMS *ID fields that do not change their value*. An ID field:
  - Is used with any non-unique IMS key field to uniquely identify a segment occurrence
  - Does not change its value
  - Is not defined in the IMS DBD as the IMS key field and is not part of the IMS key field

  An ID field is part of the “candidate key” of the IMS segment.

  Mapping of ID fields of a parent/ancestor as PATH data is similar to mapping the IMS key field of a parent/ancestor, and does not result in denormalization.

The following two sections explain more about uses of PATH data.

PATH=DENORM: Denormalizing data to improve performance of DB2 queries

Determining whether to denormalize data depends on the state of your data and how your system is used (see “Normalizing data” on page 72). If you use the DB2 copy of your data only in read-only mode (for example, for decision support

purposes) you can intentionally denormalize it to increase performance of your DB2 queries. Intentional denormalization often avoids the performance overhead of joins during queries.

The example in Figure 7 on page 30 assumes that you need to propagate an employee database consisting of three segment types:
- EMPLOYEE
- SKILLS
- AWARDS

In this case, the name of the employee is stored in the EMPLOYEE segment.

The three segments are propagated with three mapping case 1 propagation requests to the three tables EMPLOYEE, SKILLS, and AWARDS. Assume that almost all DB2 queries of the SKILLS or AWARDS table need to include the employee name in their output. You can then decide to include NAME columns in the SKILLS and AWARDS tables. And the NAME field of the EMPLOYEE segment can be included as PATH data in the mapping of those propagation requests that do propagation to the SKILLS and AWARDS tables. You reduce the number of queries to SKILLS and AWARDS tables that access the EMPLOYEE table to get the name. Access to the EMPLOYEE table is either through an explicit SQL join or through use of DB2 views that implicitly join the EMPLOYEE table to the SKILLS/AWARDS tables. Reducing the number of times the EMPLOYEE table is accessed improves the query performance.

If you use the DB2 copy only for queries and not synchronous propagation or updates to DB2 tables, the denormalization that occurs is not a problem. But if the SKILLS table, including its NAME column, can be updated through SQL, denormalization usually results in inconsistencies.

Do not denormalize data unless the PATH data is updated infrequently and queried often. Even though propagation of PATH data can improve the performance of DB2 queries, it usually decreases the performance of propagation because an update of the NAME field in the EMPLOYEE segment is usually propagated to:
- One row of the EMPLOYEE table
- Multiple rows of the SKILLS table
- Multiple rows of the AWARDS table

IMS DPROP tries to limit negative performance impact. When the parent/ancestor containing PATH data is replaced, IMS DPROP first checks whether the fields used as PATH data have changed. If the fields have not changed, IMS DPROP does not issue SQL statements to propagate the PATH data to the target table.

Figure 7 on page 30 illustrates denormalization of data. In the figure, PR2 belongs to mapping case 1 and propagates the entity segment SKILLS to the table SKILLS. PR2 includes in its mapping as PATH data the NAME field from the physical parent.

PR3 also includes in its mapping as PATH data the NAME field from the physical parent of the entity segment AWARDS.

In Figure 7 on page 30, including PATH data denormalizes the DB2 copy of the data in order to improve performance of DB2 queries.

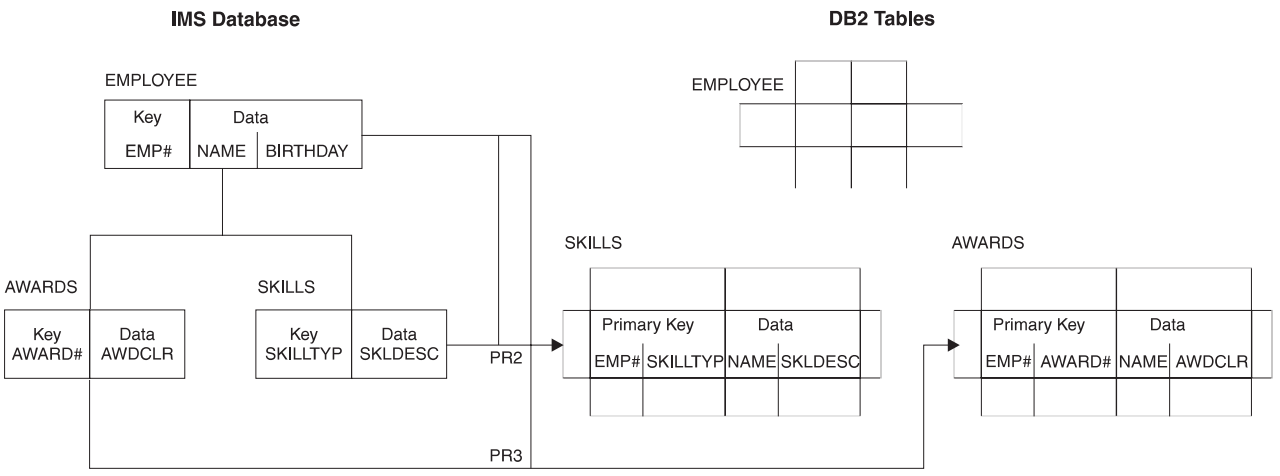


Figure 7. Denormalization of data with PATH data

Rules for PATH=DENORM

1. A propagation request can be defined only as PRTYPE=L. Only one-way IMS-to-DB2 propagation is supported.
2. Fields included in PATH data can change their values. Your IMS application programs can change the value of PATH data with REPL calls. If the value of PATH data can change, ensure that the following rules are observed when defining your propagation request:
   a. If you are defining matching DB2 RIRs, as explained in "DB2 referential integrity guidelines" on page 45, do not map PATH data fields that can change their value to a:
      - DB2 foreign key
      - DB2 primary key, if the target row has dependent rows

      When creating the propagation requests, IMS DPROP writes warning messages if it detects PATH data mapped to a foreign or primary key. Propagation can fail.

   b. Uniquely identify each parent/ancestor segment contributing modifiable PATH data to the propagation request. Identify segments by combining fields[2] mapped by the propagation request and located in either:
      - The parent/ancestor
      - A segment higher in the hierarchy

      Figure 8 on page 31 shows parent/ancestor with PATH data. Both the parent SEG2 and the ancestor SEG1 contribute modifiable PATH data to PR3. The unique identification rule applies to each parent/ancestor segment contributing PATH data. The rule requires that:

      - SEG2 occurrences be uniquely identified through the combination of those mapped fields located in SEG2 or in a segment higher in the hierarchy, such as SEG1. Therefore, SEG2 must be uniquely identified through a combination of DATA2, KEY2, DATA1, and KEY1 values.

2. IMS-to-DB2 mapping of those fields that uniquely identify the changed parent/ancestor should not cause loss of uniqueness.

- SEG1 occurrences be uniquely identified through a combination of DATA1 and KEY1 values.

  IMS DPROP does not check for rule compliance. If you violate the rule, IMS DPROP propagates changes of the PATH data to the wrong DB2 rows, resulting in data inconsistency.

  In the example in Figure 8, rule violation can cause updates of PATH data in one occurrence of SEG2 or SEG1 to be propagated to the wrong TAB3 rows. For example, the PATH data would go to SEG3 segments, which are dependents of other SEG2 or SEG1 parent/ancestors.

  If all parent/ancestors contributing to PATH data have unique IMS fully concatenated keys and if their whole IMS fully concatenated key is included in the mapping, you have complied with the rule.

3. Do not map PATH data to a DB2 LONG VARCHAR column longer than 254 characters or a LONG VARGRAPHIC column longer than 127 DBCS characters.
4. Do not map PATH data to a DB2 column that can contain a null value if the propagation of an IMS change can result in a DB2 null value. For example:
   - Do not map a PATH field of a variable-length segment if the PATH field is not in the existing part of each segment occurrence.
   - Do not map a field processed by a Field exit routine that requests mapping to a DB2 null value.
5. Do not map a PATH field of a variable-length segment to a DB2 DATE, TIME, or TIMESTAMP column if the PATH field is not in the existing part of each segment occurrence.

Identifying Parent/Ancestors Contributing Modifiable PATH Data to PR3

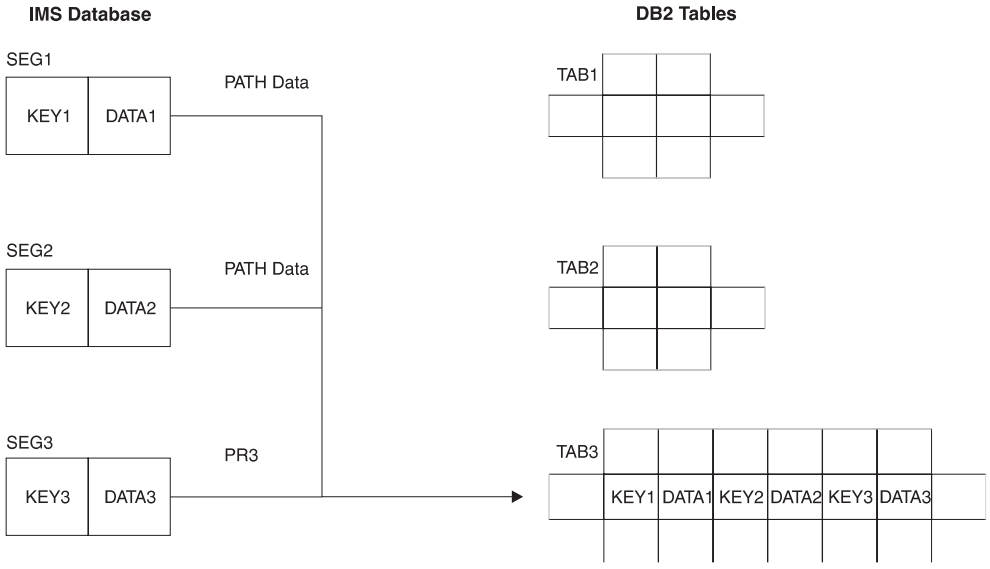


Figure 8. Identifying Parent/ancestors contributing modifiable PATH data to PR3

PATH=ID: Mapping ID fields of a physical parent/ancestor

Sometimes, IMS segments are defined in the DBD without a key field or with a non-unique key field, even if they have unique candidate keys. An example is when IMS applications need to retrieve the segment in a sequence other than the ascending sequence of the candidate segment key.

When propagating dependents of such a segment, you usually propagate the candidate key of the parent/ancestor to DB2 in the same way you would if the candidate key was an IMS key field. To do so, you propagate the candidate key of the parent/ancestor as PATH data to the foreign key of the target table.

Including ID fields of a parent/ancestor in the mapping does not denormalize your DB2 data copy. Instead, it results in a clean normalized DB2 data structure with proper DB2 primary and foreign keys. Because ID fields are part of a candidate key, their values should not change.

In the example shown in Figure 9 on page 33, the IMS database consists of three segment types: SEG1, SEG2, and SEG3. SEG2 is defined in the DBD without an IMS key field but does have a candidate key, ID2. When performing the DB2 table design, you want to include ID2 in the primary key of TAB2. You also want to include ID2 in both the primary and foreign key of TAB3; you can do this by including ID2 as PATH data in the mapping of PR3, which propagates to TAB3.

In this case, including ID2 in the TAB3 table and in the PATH data of PR3 does not result in denormalizing DB2 data. Instead, you implement a clean DB2 table design with proper DB2 primary and foreign keys.

In this example, the field included as PATH data (ID2) is a candidate key that never changes its value in either the IMS or DB2 data copy.

Rules for PATH=ID

1. A propagation request can be defined as either PRTYPE=E or L. If defined as PRTYPE=E, it can support:
   - One-way IMS-to-DB2 propagation
   - One-way DB2-to-IMS synchronous propagation
   - Two-way synchronous propagation
2. You can include as PATH data only ID fields that do not change their value.

   You cannot change the value of ID fields through either DL/I replace calls or SQL update calls because propagation will fail.

   If you need to change the value of an ID field, consider deleting the current occurrence of the segment or row and reinserting a new occurrence with the changed value. You may need to change your applications.
3. For PRTYPE=E, ID fields included in PATH data are subject to the IMS DPROP key mapping rules explained in “Rules For PRTYPE=E (Extended Function)” on page 53.

   You must map the ID fields used as PATH data to the primary DB2 key.

   All PRTYPE=Es in a propagation request set that propagate a particular segment (or the dependents of that segment) must map the ID fields of the segment to the DB2 primary key of their respective target tables.

   In Figure 9 on page 33, all propagation requests propagating SEG2 and all propagation requests propagating its dependent SEG3 must map ID field ID2 of SEG2 to the DB2 primary key of their respective tables.

In Figure 9, PR3 belongs to mapping case 1. PR3 propagates entity segment SEG3 to table TAB3. The ID field ID2 of the physical parent segment SEG2 is propagated as PATH data exactly the same as if ID2 had been the IMS key field of segment SEG2.

PR Propagating ID Fields of a Physical Parent/Ancestor as PATH Data

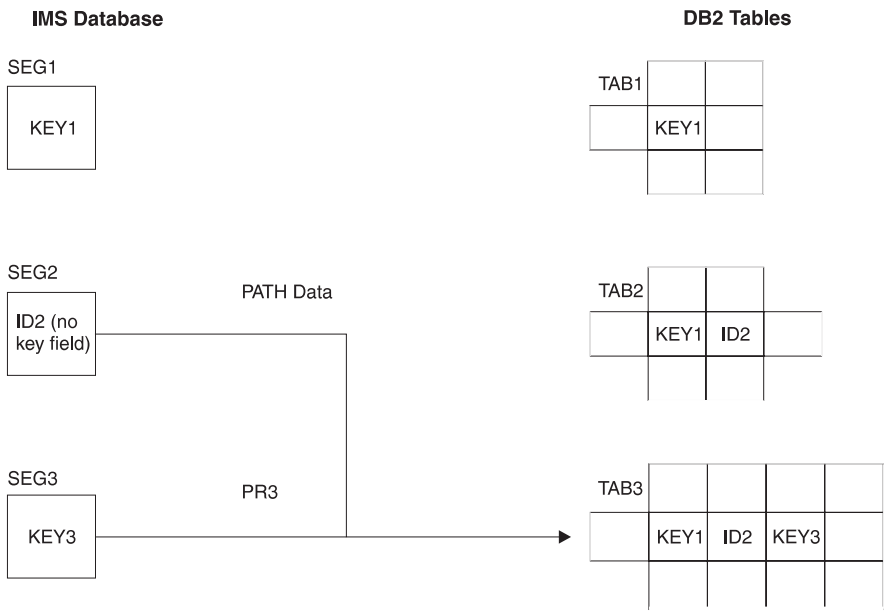


Figure 9. PR propagating ID fields of a physical parent/ancestor as PATH data

WHERE clause

You can specify a WHERE clause in the propagation request for mapping cases 1 and 2, but not 3. The WHERE clause specifies under which conditions a segment occurrence is to be propagated from IMS to DB2. The conditions are based on field values or a combination of field values.

By defining multiple propagation requests with different WHERE clauses, you can propagate the same segment type to or from different tables. You can propagate segment types containing different kinds of data, for example, redefined data to or from different tables.

Figure 10 on page 34 shows mapping with a WHERE clause.

When propagating an IMS REPL operation for a propagation request specifying a WHERE clause, the RUP needs to evaluate the WHERE clause both for the “before replace image” and “after replace image.” Depending on these evaluations, RUP either issues SQL UPDATE, DELETE, or INSERT statements or does nothing.

In Figure 10 on page 34, segment SEG2 contains redefined data. In this example, SEG2 can contain two different kinds of data. The kind of data in a particular occurrence of SEG2 is identified by the value of field F2, which can have the following values: ’A’, ’a’, ’B’, and ’b’.

The two kinds of data are propagated by two different mapping case 1 propagation requests to two different tables:

- PR2A is defined with a WHERE clause specifying F2='A' or F2='a'. PR2A propagates to TAB2A those occurrences of SEG2 that contain the value 'A' or 'a' in field F2.
- PR2B is defined with a WHERE clause specifying F2='B' or F2='b'. PR2B propagates to TAB2B those occurrences of SEG2 that contain the value 'B' or 'b' in field F2.

The WHERE clause has a very limited use in DB2-to-IMS propagation. If propagation is from DB2-to-IMS, the arrows in Figure 10 would simply be reversed.

Mapping with a WHERE Clause

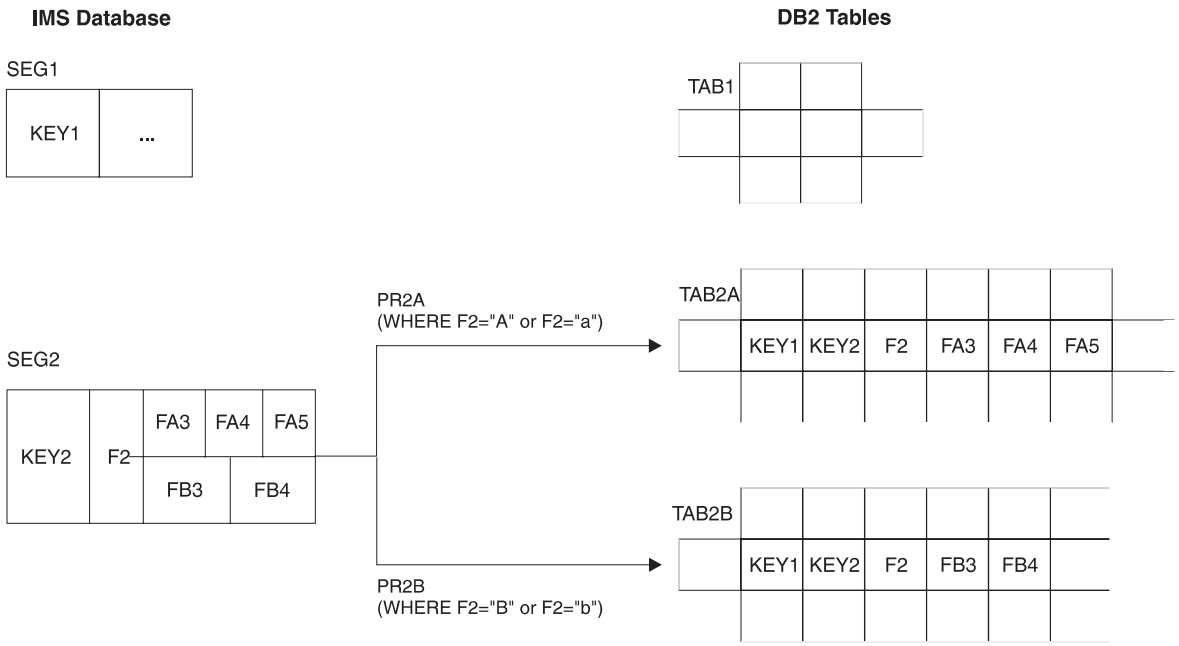


Figure 10. Mapping with a WHERE clause

The following sections discuss:

- Selective propagation
- Fields that can be included in the WHERE clause
- Fields that cannot be included in the WHERE clause
- Conditions and operators
- Propagating parent segments
- Propagating logical parent segments

Selective propagation using the WHERE clause

For PRTYPE=E, the WHERE clause is supported for both IMS-to-DB2 propagation and DB2-to-IMS synchronous propagation. However, IMS DPROP support of the WHERE clause for DB2-to-IMS synchronous propagation is different from the support for IMS-to-DB2 propagation.

The WHERE clause allows only selective propagation from IMS-to-DB2. You specify the conditions under which a segment occurrence is to be propagated from IMS-to-DB2. Each specified condition compares the value of an IMS field with the value of another IMS field or a literal. The WHERE clause permits IMS DPROP to work with unorthodox IMS database designs that use the same IMS segment type to store different kinds of information.

You cannot use the WHERE clause to compare the value of DB2 columns with the value of other DB2 columns or literals.

You might store some rows that you do not want to synchronously propagate with other data in a propagated table. For example, you have rows that contain data that existing IMS applications are not prepared to handle. You can use a segment exit routine to selectively suppress propagation to the segment. The segment exit routine runs after IMS DPROP maps the row into the defined segment format.

Fields that can be included in the WHERE clause

In the WHERE clause, you can include:

- IMS *key fields* or subfields of keys from each segment in the physical hierarchical path of the entity segment, from the entity segment up to the root.
- Fields belonging to *PATH data* in the parent or in an ancestor of the entity segment, when the propagation request specifies PATH=ID. These are fields that cannot change their value. IMS DPROP stops the creation of propagation requests if the WHERE clause includes PATH data and PATH=DENORM.
- The following *non-key fields of the entity segment*:
  - For the *lowest* propagated entity segment in each hierarchical path, you can also include non-key fields of the entity segment except for mapping case 2. You cannot include non-key fields that can change their value.

    When running propagation requests that belong to multiple propagation request sets, you propagate the same data to multiple sets of DB2 tables.
  - For an entity segment that is *not the lowest* propagated segment in its hierarchical path, the following rules apply:
    1. For PRTYPE=E or if you are implementing DB2 RIRs matching the IMS parent/child relationships, include in the WHERE clause only fields of the entity segment that do not change their value[3].
    2. For PRTYPE=L without matching DB2 RIRs, include in the WHERE clause any field of the entity segment except for mapping case 2, which cannot include fields of the entity segment that can change their value.

Fields which cannot be included in the WHERE clause

You *cannot* include these fields in the WHERE clause:

- Fields in dependents of the entity segment (for example, fields located in an extension segment of mapping case 2). IMS DPROP checks for this when you define a propagation request.
- PATH data fields if the propagation request specifies PATH=DENORM, meaning the PATH data fields can change their value. IMS DPROP checks for this when you define a propagation request.

3. MVG writes warning messages when it detects non-key fields in the WHERE clause. But only for segments other than the lowest propagated segment in a path. The message text or description tells you this is not a problem if the field cannot change its value.

- For mapping case 2, fields located in the entity segment that can change their value[4]. IMS DPROP checks for this when you update the entity segment or target row.

IMS DPROP has the following additional requirements for PRTYPE=E defined with a WHERE clause:

- You must map all fields of the entity segment included in the WHERE clause to the target table and map all fields of the IMS fully concatenated key to the target table.
- You can propagate a segment type using multiple PRTYPE=E with different WHERE clauses. However, you should propagate one particular segment occurrence with only one PRTYPE=E. IMS DPROP checks this at propagation time.

Conditions and operators used with the WHERE clause

IMS DPROP and DataRefresher support use of multiple conditions in a WHERE clause. You can combine multiple conditions with the following operators:

AND
OR

When providing multiple conditions, you can control the priority of conditions by using parentheses.

IMS DPROP and DataRefresher support comparisons using the following operators:

=
>
>=
<
<=
¬=

Unlike DataRefresher, IMS DPROP does not support the following operators:

NOT
LIKE
NOT LIKE
IN
NOT IN
BETWEEN

Refer to the *Reference* for more details on what you can specify in a WHERE clause.

Recommendations for propagating parent segments with a WHERE clause

With a WHERE clause, the physical or logical dependents of parent segment types are propagated under the same conditions as the parent. If you implement DB2 RIRs and do not use the same WHERE clauses for propagation of the parent and dependents, you risk propagation failures. Depending on your propagation request definitions, propagation could fail because of RIR violations. For example, IMS DPROP tries to propagate the insert of a dependent segment, and its physical or logical parent has not been propagated.

4. MVG writes warnings when it detects a non-key field of a mapping case 2 entity in a WHERE clause.

IMS DPROP checks each propagation request set to see if the WHERE clause specified for propagated parents and dependents is identical. If not, IMS DPROP writes warning messages. You can choose to ignore warning messages and continue to propagate dependents separate from their parents. You must determine whether the different WHERE clauses are acceptable.

Recommendation for propagating logical parent segments with a WHERE clause

Include in the WHERE clause for a logical parent segment only IMS key fields (or subfields of keys) from each segment in the physical hierarchical path of the logical parent, from the logical parent up to the root. You usually want to propagate the logical child with the same WHERE clause as the logical parent.

You cannot include either:

- Non-key fields of the logical parent
- PATH data of the logical parent

in the WHERE clause because IMS DPROP only knows the logical parent's fully concatenated key.

Chapter 3. Propagation guidelines, rules, and restrictions

This section presents guidelines and rules for preparing, mapping and designing propagation. The information provided is *not* at a step-by-step task level, but is presented in the order in which tasks should be done and rules should be considered.

Topics included are:

- Propagation guidelines for segments, tables, rows and fields
- The rules and recommendations for defining DB2 referential integrity relationships between propagated tables
- The rules and recommendations for defining unique IMS secondary indexes and unique DB2 indexes
- The rules for mapping between IMS and DB2 keys
- The field formats and field conversions supported by IMS DPROP
- Normalizing data

Propagation guidelines

As you map your data and define propagation, follow the recommendations and rules presented in this section. Topics include:

- IMS database options (especially those for the delete rules for logical relationships)
- The DB2 primary key for tables, used with the three mapping cases provided by IMS DPROP
- Propagating variable-length segments
- Propagating a subset of fields and columns
- Mapping a field to multiple columns and mapping multiple fields to a column
- Exceptions to the typical implementations of:
  - One-to-one mapping between fields and columns
  - Propagation of a given table with a single propagation request
  - One segment type being propagated to or from only one table
- Using propagation request sets
- Defining propagation requests with qualified or unqualified table names

IMS logical relationship rules

This section describes the rules for propagating segments involved in logical relationships.

Paired logical children

- If you have IMS logical relationships with paired logical children, only one of the pair should be propagated:
  - If the pairing is virtual, then the physical child should be propagated.
  - If the pairing is physical, then the child with propagated physical dependent segments should be propagated.

  If you do not observe these rules for generalized mapping cases, IMS DPROP either writes warning messages when creating the propagation request (PRTYPE=L and F) or does not create the propagation request (PRTYPE=E). (For an explanation of types of propagation requests, see “Selecting a propagation request type” on page 12.)

Delete rules

As shown in Table 3, some delete rules for IMS logical relationships are not supported by IMS DPROP and the IMS Data Capture function.

If a segment involved in a logical relationship does not use one of the supported delete rules, change the DBD so that it does. You may also need to change application programs that use the propagated database.

Table 3. Supported IMS delete rules. This table shows the IMS delete rules supported by IMS DPROP for segments involved in logical relationships. X=delete rule is supported.

Segment	Which IMS delete rule is supported?			
	VIRTUAL	PHYSICAL	LOGICAL	BIDIRECTIONAL VIRTUAL
Logical child	X			
Logical parent involved in bidirectional IMS relationship		X	X	
Logical parent involved in unidirectional IMS relationship		X	X (See following description)	
Physical parent (of a logical child)	X	X	X	

Logical children: Logical child segments involved in propagation must have an IMS delete rule of VIRTUAL. The physical and logical parents and ancestors of a logical child involved in propagation also cannot be propagated unless the logical child has a delete rule of VIRTUAL.

Logical parents: Logical parent segments must have a delete rule of either PHYSICAL or LOGICAL.

A delete rule of PHYSICAL requires that you delete all logical children before deleting the logical parent.

A delete rule of LOGICAL allows you to delete a logical parent even when it has existing logical children. Deletion of a logical parent prevents further access to the logical parent from a physical path, but not from a logical path. The logical parent, and any of its physical ancestors, remain accessible from any existing logical children. For Unidirectional relationships, IMS DPROP generalized mapping logic usually requires a delete rule of PHYSICAL.

Use a delete rule of LOGICAL only with user mapping or if you implement one-way IMS-to-DB2 propagation with PRTYPE=L (PRTYPE=L are described in “Selecting a propagation request type” on page 12). A delete rule of PHYSICAL is preferable to LOGICAL because with a LOGICAL delete rule:

- IMS DPROP generalized mapping logic propagates a delete of the logical parent and any of its physical ancestors, even if the segment remains accessible from a logical child through a logical path. Therefore, you might be able to access an IMS segment through a logical path even though the corresponding DB2 row no longer exists.

  Retrieving the logical parent segment or its ancestor through a physical path should provide the same results as accessing a DB2 row.

- You cannot establish a matching DB2 referential integrity relationship between the target table of the logical parent and the target of the logical child. If you implement a DB2 referential integrity relationship between the targets of a logical parent and logical child, propagation usually fails.

For user mapping logic with a delete rule of LOGICAL, your Propagation exit routines must decide whether to delete the target DB2 row:

- As soon as the IMS segment is deleted on the physical path, even if the segment remains accessible through a logical path
- Only when the IMS segment is both physically and logically deleted

Physical parents: The physical parent of a logical child must have either a VIRTUAL, PHYSICAL, or LOGICAL delete rule.

The IMS delete rule for a physical parent has meaning only for logical relationships implemented with virtual pairing.

Requirement for a DB2 primary key

For the generalized mapping cases, IMS DPROP requires that propagated DB2 tables have a primary key even if you do not implement RIRs between propagated tables.

The following topics describe constraints your IMS databases must conform to in order to be propagated using IMS DPROP.

Propagating variable-length segments (IMS-to-DB2)

IMS DPROP requires that every field be either completely contained within or completely absent from the segment occurrence it is propagating. Because segment length varies, the start or end position of a particular field mapped by a propagation request might extend beyond the end position of the segment occurrence, causing a discrepancy between the application program's use of the data field and the mapping definition of the data field. You must be careful when mapping the fields of IMS variable-length segments.

If the field is absent from the segment occurrence, IMS DPROP maps the field to either:

- A null value, if the target column permits
- The default value, if the target column is defined as NOT NULL WITH DEFAULT

If the target column is defined as NOT NULL, propagation fails. See the example shown in Figure 11.

Defining Variable-Length Segments

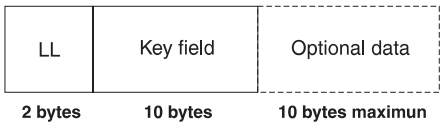


Figure 11. Defining variable-length segments

The segment could be defined to IMS as minimum length 12 bytes, maximum length 22 bytes, with the key in position 3-12. Assume that the same segment is defined to IMS DPROP as two fixed-length fields, *key* and *data*, each 10-byte character fields.

The application program that manipulates this segment can insert or replace a segment occurrence so the data field is either:

- Completely contained within the segment occurrence (the segment length being 22 bytes)
- Completely absent (the segment length being 12 bytes)
- Shorter than the 10 bytes expected by IMS DPROP (the segment being greater than 12 bytes and less than 22 bytes), which is not valid for IMS DPROP

Propagating a subset of columns in a table

For IMS-to-DB2 propagation, during insert operations, propagation sets the values of the nonpropagated columns to a default or a null value.

This section describes:

- How nonpropagated columns are handled by IMS DPROP during IMS-to-DB2 propagation
- Requirements and recommendations for one-way IMS-to-DB2 propagation

Assigning values to nonpropagated columns during IMS-to-DB2 propagation:
During IMS-to-DB2 propagation:

- When *replacing* a row, IMS DPROP does not change the value of columns that are not propagated.
- When *inserting* a row, IMS DPROP does not set any value in nonpropagated columns. Therefore:
  - If the nonpropagated column permits a null value, DB2 sets it to null.
  - If the nonpropagated column is defined as NOT NULL WITH DEFAULT, DB2 sets it to the default value for its data type.

Requirements and recommendations for one-way IMS-to-DB2 propagation:
Usually all columns in a table are propagated. Columns that are not propagated must either permit a null value or be defined as NOT NULL WITH DEFAULT.

When doing one-way IMS-to-DB2 propagation, map all columns in a propagated table to make it easier to recreate the DB2 data copy based on the IMS data copy:

- If you recreate the DB2 tables with DataRefresher, columns that are not propagated are set either to a null value or to their default value. You must then provide DB2 update programs that reconstruct the real value of these columns.
- If you use CCU-generated DB2 repair statements, insert repair statements do not set any value in nonpropagated columns. Columns that are not propagated are set either to a null value or to their default value. You must then provide DB2 update programs that reconstruct the real value of these columns.
- Nonpropagated columns can be updated only by your updating SQL statements.

Make sure your SQL statements do not update propagated columns and do not delete or insert the whole row. With one-way IMS-to-DB2 propagation, SQL updates are not propagated to the IMS copy and, therefore, jeopardize consistency between the DB2 and IMS copy. Consider using DB2 security to prevent SQL statements

from updating propagated columns or inserting and deleting rows. See “Updates to non-propagated columns” on page 122 for more information on how you can use DB2 security.

Mapping between fields and columns

Usually, you implement a one-to-one mapping between fields and columns so that one field propagates to only one column, and one column propagates from only one field.

Mapping one field to multiple columns

- With PRTYPE=L, you can map a propagated field to more than one column in the same table.
- With PRTYPE=E, you cannot map a field or part of a field to multiple DB2 columns if the field is part of the IMS key or is mapped to the DB2 primary key.

  Even though IMS DPROP allows you to map other fields or parts of other fields to multiple columns, avoid doing so. During DB2-to-IMS synchronous propagation, you might create inconsistencies.

Mapping multiple fields to one column

IMS DPROP generalized mapping does not support mapping multiple fields to one column.

Propagating with multiple propagation requests to the same table

When using the generalized mapping cases, you can only propagate with a single propagation request to or from a given DB2 table. You can propagate with multiple propagation requests to or from the same table using a user mapping case but you cannot propagate to or from the same table with both user mapping and generalized mapping cases.

Propagating one segment to multiple tables

Usually you propagate one segment type to or from only one table. But, following the rules described in this section, you can also propagate one segment type to or from multiple DB2 tables by creating multiple propagation requests for it.

PRTYPE=L and one-way IMS-to-DB2 propagation

Implement one-way IMS-to-DB2 propagation with PRTYPE=L. You can propagate one segment to multiple tables.

IMS DPROP allows you to propagate the same segment with a PRTYPE=E and one or multiple PRTYPE=Ls.

PRTYPE=U

IMS segments propagated exclusively through user mapping (PRTYPE=U) can be propagated to or from multiple tables.

Using propagation request sets

When you define a propagation request, you can specify an eight-byte propagation request set identifier (PRSET ID).IMS DPROP records the PRSET ID of each propagation request in the propagation request table in the IMS DPROP directory. All propagation requests with the same PRSET ID are considered part of the same propagation request set.

Sometimes it is convenient to group logically related propagation requests into the same propagation request set.[5] A number of IMS DPROP control statements support a PRSET= keyword. When you use the PRSET= keyword, IMS DPROP applies the control statement to the propagation requests belonging to the specified propagation request set. You might find it more convenient to specify a PRSET ID on IMS DPROP control statements than to specify a long list of individual propagation request IDs.

Each execution of the CCU processes only propagation requests belonging to one propagation request set.

Examples of propagation request set use

Examples of using propagation request sets are:

Example 1: You usually propagate the same IMS data to only one set of DB2 tables. However, you might want to propagate the same IMS segments to multiple DB2 tables. Perhaps you want to propagate one IMS database using one set of propagation requests to one set of tables used for operational applications, and propagate the same IMS database using a second set of propagation requests to a second set of tables used for decision support.

You can use:

- One PRSET ID for the propagation requests propagating to the first set of tables
- Another PRSET ID for the propagation requests propagating to the second set of tables

You can document the propagation requests that belong together and have a better overall view of your propagation request definitions. You also simplify your use of IMS DPROP utilities, such as the SCU and CCU. When providing utility control statements, you do not need to specify long lists of table names or propagation request IDs. Instead, you can specify a PRSET ID.

Example 2: If you propagate the data of two different applications that have their own distinct groups of databases, you can use:

- One PRSET ID for the propagation requests propagating the databases of the first application
- Another PRSET ID for the propagation requests propagating the databases of the second application

Defining propagation requests with qualified or unqualified table names

When defining a propagation request, you specify the name of the propagated DB2 table. You can specify either a qualified table name (a two-part table name) or an unqualified table name (a one-part table name).

Qualified table names

A propagation request definition with a qualified table name supports propagation to or from only one table, whose qualified name is defined in the propagation request.

IMS-to-DB2 propagation: During propagation request definition, if you specify a qualified table name, the SQL statements in the SQL update module that has been generated use the specified qualified table name. Therefore, the propagation

5. In contrast to IMS DPROP R1, IMS DPROP R2 does not require that you group propagation requests into multiple propagation request sets based on DB2 referential integrity structures.

request propagates to the same qualified table name; it cannot propagate to other identically structured tables with other table name qualifiers.

Unqualified table names

A propagation request definition with an unqualified table name can support propagation to or from one of multiple, identically structured tables that have the specified unqualified table name.

IMS-to-DB2 propagation: During propagation request definition, if you specify an unqualified table name, the SQL statements in the SQL update module that has been generated use the specified unqualified table name. Therefore, you can use the propagation request to propagate to any table that has:

- The unqualified name specified during propagation request definition
- The same structure as a model table identified during propagation request definition

When binding a package or the plan of the propagating application, the BIND process sets the qualifier, which determines the qualified table name of the propagated table. If you are using the bind package function, use the QUALIFIER= keyword of the BIND PACKAGE command to set the qualifier. If you are not using the bind package function, setting the qualifier is more complex; for more information on this subject, refer to “DB2 ALIAS and SYNONYM statements” on page 133.

You may also bind multiple DB2 packages or plans for the same application so that each bind sets a different qualifier. For example, if you specified TABLE01 as an unqualified table name during propagation request definition, you can then bind a first package or plan so that the BIND process sets SANDY as a qualifier. You can then bind another package or plan so that the BIND process sets HOWARD as another qualifier. Then, depending on which DB2 package or plan you use for the propagating application, the propagation request propagates either to table SANDY.TABLE01 or table HOWARD.TABLE01.

Binding multiple packages or plans is useful in some test environments where Sandy and Howard have their own identically structured copy of TABLE01. Sandy and Howard may share the same propagation request. You do not need to define the propagation request again for each copy of TABLE01. Using the same propagation request, Sandy’s tests propagate to SANDY.TABLE01, while Howard’s tests propagate to HOWARD.TABLE01.

A propagation request with one DB2 package or plan can propagate to only one particular table. However, processing the same propagation request with another DB2 package or plan allows propagation to another table.

DB2 referential integrity guidelines

For background information on DB2 referential integrity, refer to *DB2 Administration Guide*. This book uses the term *referential integrity relationship* (RIR) instead of the DB2 term *referential integrity constraint* to emphasize the similarity between IMS parent/child relationships and DB2 referential integrity parent/child relationships.

If you are using a generalized mapping case, there are rules for implementing DB2 RIRs involving propagated tables. By following these rules, you avoid failures in the propagation of IMS updates (IMS-to-DB2).

When you create propagation requests for a generalized mapping case, IMS DPROP does only limited checking for RIRs that are incompatible with propagation. For more comprehensive checking, use the MVGU revalidation function after all propagation requests are created.

Implementation of RIRs involving propagated tables is *optional*. If implemented, the DB2 parent/child RIRs should be a matching *subset* of the IMS parent/child relationships.

If RIRs are not a matching subset, propagation fails. The DB2 referential integrity requirements do not match any equivalent IMS parent/child relationship.

Installations that do one-way IMS-to-DB2 propagation usually do not implement DB2 RIRs for the propagated tables because DB2 tables are not usually updated through SQL and, therefore, do not need DB2 referential integrity constraints for SQL updates.

Defining DB2 RIRs to match IMS relationships

To define a DB2 parent/child RIR that matches an IMS parent/child relationship, use the following rules:

1. The corresponding table for the parent segment must be designated as the parent table.
2. The corresponding table for the child segment becomes the child table.
3. The primary key of the *parent* table and the foreign key of the *child* table should be mapped from the same IMS fields.

   Ideally, the primary key of the parent table and the foreign key of the child table are the mapped IMS fully concatenated key of either the:

   - Physical parent segment, if matching a physical IMS parent/child relationship
   - Logical parent segment, if matching a logical IMS parent/child relationship

   This ideal case requires that the parent segment have a unique IMS fully concatenated key.

   Exceptions to the ideal case are:

   - For PRTYPE=E and for a DB2 RIR matching a *physical* IMS parent/child relationship: if the parent segment has no unique IMS fully concatenated key but has a unique conceptual fully concatenated key, then the primary key of the parent table and the foreign key of the child table are the mapped conceptual fully concatenated key of the physical parent segment.

     The conceptual fully concatenated key is defined in “Terminology related to keys” on page 49.
   - For PRTYPE=L, the rules are less strict than for PRTYPE=E. For a DB2 RIR matching a *physical* IMS parent/child relationship: the primary key of the parent table and the foreign key of the child table should be mapped from the same combination of fields located in the:
     - IMS fully concatenated key of the physical parent segment.
     - Data portion of the physical parent or physical ancestor segment. These fields should not change their value.
   - For a DB2 RIR matching a *logical* IMS parent/child relationship: the primary key of the parent table and the foreign key of the child table should be mapped from the same combination of fields located in the IMS fully concatenated key of the logical parent segment.

Using DB2 delete rules for matching RIRs

This section describes DB2 delete rules you can use to implement RIRs matching IMS parent child/relationships. Valid DB2 delete rules for RIRs matching IMS *physical* parent/child relationships are different from those matching IMS *logical* parent/child relationships.

RIR matching a physical IMS parent/child relationship

When a DB2 RIR matches a *physical* parent/child relationship, the applicable DB2 delete rule can be ON DELETE CASCADE or ON DELETE RESTRICT.

- ON DELETE CASCADE is always valid and does not cause propagation to fail.
- Use ON DELETE RESTRICT only if you have specified or defaulted to (CASCADE,KEY,DATA) on the EXIT keyword of the IMS DBD. If you are mapping PATH data to the DB2 primary key, specify the following on the EXIT keyword:

  ```
  (CASCADE,KEY,DATA,PATH)
  ```

ON DELETE SET NULLS cannot be used to match IMS parent/child relationships.

RIR matching a logical IMS parent/child relationship

When a DB2 RIR matches a *logical* parent/child relationship, the applicable DB2 delete rule depends on the delete rule for the IMS logical parent.

- An IMS delete rule of PHYSICAL for the logical parent, which requires that all logical child segments be deleted before the logical parent is deleted, is usually matched by ON DELETE RESTRICT. The matching applies to both PRTYPE=E and PRTYPE=L.

  For PRTYPE=L, you can also match an IMS delete rule of PHYSICAL with a DB2 rule of ON DELETE CASCADE.
- An IMS delete rule of LOGICAL for a logical parent involved in a bidirectional relationship is matched by ON DELETE CASCADE for PRTYPE=E and PRTYPE=L.

  An IMS delete rule of LOGICAL for a logical parent involved in a unidirectional IMS relationship is supported for one-way IMS-to-DB2 propagation with PRTYPE=L but cannot be matched by any DB2 RIRs. Do not implement an RIR between the targets of the logical parent and logical child because propagation will fail.
- IMS DPROP and the IMS Data Capture function do not support an IMS delete rule of VIRTUAL for a logical parent.

The valid combinations of IMS delete rules for logical parents and DB2 delete rules for matching RIRs are described in Table 4. The valid combinations are different for PRTYPE=L, which supports only IMS-to-DB2 propagation, and PRTYPE=E, which supports both IMS-to-DB2 propagation and DB2-to-IMS synchronous propagation.

Table 4. Valid combinations of delete rules for DB2 RIRs matching IMS logical parent/child relationships (Generalized mapping cases)

IMS Delete Rule for Logical Parent	Type of IMS Logical Relationship	Matching DB2 Delete Rule	Notes
PRTYPE=E supporting IMS-to-DB2 propagation and DB2-to-IMS synchronous propagation			
Physical	Unidirectional or Bidirectional	Restrict	Logical children must be deleted before the logical parent (IMS-to-DB2 propagation).

Table 4. Valid combinations of delete rules for DB2 RIRs matching IMS logical parent/child relationships (Generalized mapping cases) (continued)

IMS Delete Rule for Logical Parent	Type of IMS Logical Relationship	Matching DB2 Delete Rule	Notes
Logical	Bidirectional	Cascade	Logical children do not need to be deleted before the logical parent (IMS-to-DB2 propagation).
Logical	Unidirectional	N/A	PRTYPE=E does not support a LOGICAL delete rule for unidirectional relationships.
PRTYPE=L supporting one-way IMS-to-DB2 propagation			
Physical	Unidirectional or Bidirectional	Restrict	Logical children must be deleted before the logical parent.
Physical	Unidirectional or Bidirectional	Cascade	Logical children must be deleted before the logical parent.
Logical	Bidirectional	Cascade	Logical children do not need to be deleted before the logical parent.
Logical	Unidirectional	N/A	You should not implement RIRs.

Defining DB2 RIRs for IMS-to-DB2 propagation

Implementation of DB2 RIRs is optional. To avoid propagation failures, DB2 RIRs should be a matching subset of the IMS parent/child relationships. Specifically, you can implement:

- DB2 parent/child RIRs that match the physical or logical IMS parent/child relationship between the segments that are the source of the tables.
- DB2 RIRs you should implement are DB2 parent/child RIRs between a propagated parent table and a nonpropagated child table with a DB2 delete rule of ON DELETE CASCADE or ON DELETE SET NULL.

Because an IMS segment can have only two parent segments—a physical and a logical parent— you must restrict the form and numbers of DB2 RIRs you implement to prevent propagation failures. If you choose to implement other RIRs, you risk propagation failure.

Implementing non-matching RIRs for IMS-to-DB2

You can implement DB2 RIRs that do not match existing IMS parent/child relationships. For example, if you have in your IMS databases application-managed data relationships that are not reflected by IMS parent/child relationships, you can choose to implement DB2 RIRs that match the application-managed data relationships. To guarantee that the DB2 RIRs do not result in failures during processing of the SQL statements propagating the IMS updates, you need detailed and precise knowledge of the underlying databases and the application programs used to maintain them.

Defining unique indexes

To avoid propagation failures, observe IMS IMS DPROP rules when defining:
- Unique DB2 indexes and doing IMS-to-DB2 propagation
- Unique IMS secondary indexes

Definition of *non-unique* DB2 and IMS secondary indexes does not cause propagation to fail and is not subject to IMS DPROP restrictions.

This book uses the term *truly unique* IMS secondary index because an IMS secondary index is considered unique only if the combination of fields it indexes has unique values. An IMS secondary index is not considered unique if it has been artificially made unique through use of /SX or /CK fields. /SX and /CK IMS secondary indexes do not cause propagation to fail and are not subject to IMS DPROP restrictions.

Unique DB2 indexes and one-way IMS-to-DB2 propagation

IMS-to-DB2 propagation might fail if a DB2 index enforces uniqueness not enforced by IMS. In such situations, an IMS ISRT or REPL call might not successfully propagate because the update violates the uniqueness enforced by the DB2 index.

To avoid problems, do not implement unique DB2 indexes except for:
- The index for the primary DB2 key, which must be unique
- Unique DB2 indexes that correspond to truly unique IMS secondary indexes

Consider defining non-unique DB2 indexes in cases where unique DB2 indexes create problems.

If you want to enforce uniqueness for DB2 indexes without requiring uniqueness for IMS indexes, your application programs should be able to handle any propagation failures that occur.

Key mapping rules by propagation request type

Both extended and limited function propagation requests (PRTYPE=E and L) have rules for mapping keys. There are no key mapping rules for user mapping (PRTYPE=U). This section describes key terminology and the rules for mapping IMS keys to DB2 primary and foreign keys. Topics are:
- Terminology related to keys
- Overview of the key mapping rules
- Rules for PRTYPE=E (extended function)
- Rules for PRTYPE=L (limited function)
- Comparison of key mapping rules

Terminology related to keys

This section defines IMS, IMS DPROP, and DB2 terms related to keys.

IMS key field

Usually IMS segments have a key field, although it is not required.

The IMS key field can be defined in the IMS DBD as unique or non-unique. If identified as unique, each occurrence of the segment under its physical parent has a different key field value.

The IMS key field is identified in the IMS DBD using the *SEQ* sub-parameter in the NAME keyword of the FIELD statement.

IMS fully concatenated key

For an IMS segment, the fully concatenated key consists of the:

- Key field of the segment
- Key fields of the segment's physical parent and physical ancestors

IMS returns the fully concatenated key to applications in the key feedback area of the database's PCB when the segment is accessed through the physical path.

IMS concatenated key (physical and logical)

For an IMS segment, the concatenated key consists of the IMS key fields of the segment's immediate parent and ancestors.

Unlike the IMS *fully* concatenated key, the concatenated key does not include the IMS key of the segment itself.

A logical child segment has two concatenated keys:

- The ***physical concatenated key*** is the key of the segment's *physical* parent and the keys of the physical ancestors of the physical parent.

  The physical concatenated key of a segment is identical to the fully concatenated key of the physical parent segment.
- The ***logical concatenated key*** is the key of the segment's *logical* parent and the IMS keys of the physical ancestors of the logical parent.

  The logical concatenated key of a logical child segment is identical to the fully concatenated key of the logical parent segment.

  IMS returns the logical concatenated key to applications at the beginning of the logical child segment when the logical child is accessed through the physical path.

  See Figure 14 on page 62 for an illustration of concatenated keys.

ID fields

Ideally, a propagated entity segment has a unique IMS fully concatenated key. However, IMS DPROP's generalized mapping supports propagation when segments do not meet this ideal. Identification (ID) fields and conceptual keys are useful when some of your segments:

- Have multiple occurrences under their physical parent and have no unique IMS key field
- Are uniquely identifiable under their parent through non-key fields, or through a combination of a non-unique IMS key field and non-key fields

ID fields are non-key fields that allow you to uniquely identify a segment under its physical parent and do not change their value. Sometimes you do not define the ID fields as part of the IMS key field because IMS applications need to retrieve the segment in a sequence other than the ascending sequence of the ID fields. You can use ID fields with segments.

Internal segments with more than one occurrence must be uniquely identifiable within their containing IMS segment. You can identify them using ID fields.

For PRTYPE=E, ID fields mapped to the DB2 primary key must be defined in the IMS DBD except for ID fields of internal segments.

Conceptual key

This term applies only to PRTYPE=E.

For segments that are not unique under their parent and do not have a unique IMS key but are uniquely identifiable using ID fields, the conceptual

key is a non-overlapping combination of the non-unique IMS key field and ID fields. This combination must identify the segment uniquely under its parent.

For segments having a unique IMS key field, the conceptual key and the IMS key field are identical.

The conceptual key is not explicitly defined to IMS DPROP. Instead, IMS DPROP assumes that the conceptual key is the combination of fields within the segment that are mapped to the DB2 primary key.

Conceptual fully concatenated key

This term applies only to PRTYPE=E.

The conceptual fully concatenated key of a segment is both the:

- Conceptual key of the segment
- Conceptual keys of the segment's physical parent and physical ancestors

The conceptual fully concatenated key is, therefore, the combination of:

- The IMS fully concatenated key
- ID fields of the segment that contribute to the conceptual key of the segment
- ID fields of the physical parent/ancestors that contribute to the conceptual keys of the physical parent/ancestor

The conceptual fully concatenated key is the IMS fully concatenated key you would see if the ID fields at each hierarchical level were included in the IMS key field.

IMS DPROP's conceptual fully concatenated key is useful for propagation with PRTYPE=Es of entity segments that do not have a unique IMS fully concatenated key. The conceptual fully concatenated keys allows you to support segments with a unique conceptual fully concatenated key similar to segments with a unique IMS fully concatenated key.

Conceptual concatenated key

This term applies to only PRTYPE=E.

The conceptual concatenated key of a segment is the conceptual 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 conceptual key of the segment itself.

Because IMS DPROP does not support PATH data for a logical path, IMS DPROP does not distinguish between a physical and logical conceptual concatenated key.

DB2 primary key

A DB2 primary key uniquely identifies the rows of a DB2 table. A DB2 table can have only one DB2 primary key. The DB2 primary key can consist of one or more columns.

You must define a DB2 primary key for each table propagated by IMS DPROP's generalized mapping.

DB2 foreign key

DB2 foreign keys define DB2 RIRs.

DB2 foreign keys are defined for child tables. The DB2 foreign key of the child table must match the DB2 primary key of the parent table. A child table can be involved in multiple DB2 RIRs and can, therefore, have multiple DB2 foreign keys.

Overview of key mapping rules

Basic key mapping rules are:

1. A propagated DB2 table must have a DB2 primary key. All columns of the DB2 primary key must be mapped by the propagation request.
2. Ideally, each entity segment has a unique IMS fully concatenated key that is mapped one-to-one to the DB2 primary key.

   Exceptions to the ideal case are:

   - For PRTYPE=E: if the entity segment has a unique *conceptual* fully concatenated key, then the conceptual fully concatenated key should be mapped one-to-one to the DB2 primary key.

     The conceptual fully concatenated key is a combination of the IMS fully concatenated key and ID fields of the entity segment and/or its physical parent/ancestors. ID fields are fields that contribute to uniquely identifying a segment and that cannot change their values.
   - For PRTYPE=L: the entity segment can be uniquely identifiable by combining:
     – Fields located in its IMS fully concatenated key.
     – Fields located in the data portion of the entity segment and its physical parent/ancestors. These fields can change their value unless the results of change violate optional DB2 RIRs or the fields are PATH data fields of a propagation request defined with PATH=ID.

     The combination of fields is mapped one-to-one to the DB2 primary key.

The key mapping should not cause key values to become non-unique. Make sure that any Field exit routine used to map key fields preserves the uniqueness of the keys. And avoid data conversions such as conversion between decimal fields with different scales that can result in loss of uniqueness.

Recommendations:

- When the entity segment has a unique IMS fully concatenated key, make sure the DB2 primary key of the propagated table is the propagated IMS fully concatenated key of the entity segment.
- Whenever possible, use PRTYPE=E. IMS DPROP provides complete support for PRTYPE=E, including support for DB2-to-IMS synchronous propagation.

  Use PRTYPE=E if your entity segment either has a unique IMS fully concatenated key or can be identified uniquely by combining the IMS fully concatenated key with ID fields that do not change their value.
- For optimum performance of propagation during sequential processing of HIDAM and HISAM IMS databases:
  – The DB2 index for the DB2 primary key should be a clustered index.
  – The ordering sequences of the index for the DB2 primary key and the IMS fully concatenated key should be the same.

  For HDAM and DEDBs, if possible, cluster the propagated table in the same sequence as the physical HDAM or DEDB sequence.

Some DPROPNR Version 1 Release 1 restrictions for PRTYPE=L have been eased in following releases. Even though some of the wording has changed, DPROPNR Version 1 Release 2 and IMS DPROP Version 3 Release 1 key mapping rules for PRTYPE=L are upwardly compatible with DPROPNR Version 1 Release 1 rules, with one exception. Non-key IMS fields mapped to the DB2 primary key must be

defined in the IMS DBD. The IMS name, as defined in the IMS DBD, and the IMS DPROP name, as defined in the propagation request definition, of these fields must be the same.

DPROPNR Version 2 Release 1 and following releases handle PRTYPE=F and PRTYPE=L the same way. Because IMS DPROP handles PRTYPE=L and PRTYPE=F the same way, mention of PRTYPE=L in this book implies both propagation request types. For compatibility with DPROPNR Version 1 Release 1, you can still define PRTYPE=F in IMS DPROP Version 3 Release 1.

Rules For PRTYPE=E (Extended Function)

PRTYPE=Es have strict key mapping rules but also provide more function than PRTYPE=L. Figure 12 on page 56 and Figure 13 on page 58 illustrate the key mapping rules for PRTYPE=E. Key mapping rules for PRTYPE=E are:

Key rule 1: A propagated DB2 table must have a primary key. All columns of the DB2 primary key must be mapped by the propagation request. The IMS fields that map to the DB2 primary key must result in a unique DB2 primary key.

Key rule 2: Every byte of the IMS fully concatenated key of the entity segment must be propagated to the DB2 primary key by either:

- Using each IMS key field, which is the key of the entity segment and the key of each physical ancestor, as a single field. Each field is mapped to a column of the DB2 primary key.
- Subdividing one or all IMS key fields into non-overlapping subfields. Each subfield of a key is mapped to an individual column of the DB2 primary key.

You must also propagate every byte of the logical concatenated key for a propagated logical child segment.

Key rule 3: If you are doing DB2-to-IMS synchronous propagation, do not issue SQL UPDATE statements that change the values of the DB2 primary key columns. Ideally, a propagated entity segment has a *unique IMS fully concatenated key*, meaning the entity segments and their physical parent/ancestors have either a unique IMS key field or a maximum of one occurrence under their physical parents.

Also, the DB2 primary key is the propagated IMS fully concatenated key of the entity segment, meaning the DB2 primary key is mapped by the IMS fully concatenated key of the entity segment; and the IMS fully concatenated key of the entity segment is mapped to the DB2 primary key. The fields being mapped to the DB2 primary key must not overlap.

If a propagated entity segment does not have a unique IMS fully concatenated key, then:

- The entity segment must have a *unique conceptual fully concatenated key*. The physical parent and all physical ancestors of the entity segment must also have a unique conceptual fully concatenated key. As explained in more detail on page 51, the conceptual fully concatenated key results from adding:
  - The IMS fully concatenated key
  - The ID fields of the entity segment its physical parent/ancestors

  ID fields are fields that contribute to uniquely identifying a segment.
- The DB2 primary key must be the propagated conceptual fully concatenated key of the entity segment. The DB2 primary key must be mapped by the conceptual fully concatenated key of the entity segment; and the conceptual fully

concatenated key of the entity segment must be mapped to the DB2 primary key. The fields being mapped to the DB2 primary key must not overlap.

When including ID fields in the conceptual fully concatenated key of the entity segment, observe the following rules:

1. ID fields included in the conceptual fully concatenated key must not change their value. If the value is changed as a result of updating IMS calls or SQL statements, propagation fails.

   One exception to this rule is ID fields of internal segments. Changes in the ID field of an internal segment are interpreted by IMS DPROP as a sequence of deletes and inserts of the internal segment.
2. IMS application programs should not insert segment occurrences that violate the uniqueness rule of the target DB2 primary key because propagation fails.
3. For variable-length segments, ID fields must be located in the existing part of the segment or propagation will fail.
4. The conceptual key of a particular segment type must be identical in all PRTYPE=Es. *All* PRTYPE=Es propagating a particular segment or its dependents must include the same ID fields in the conceptual key of that particular segment. And they must all map the same ID fields to the DB2 primary key of their respective target DB2 tables.

   In Figure 13 on page 58, the conceptual key of SEG2 consists of the fields SEG2ID1 and SEG2ID2. The conceptual key of SEG2 is identical in all propagation requests propagating SEG2 and its dependents: PR2 and PR3. Both propagation requests map the same conceptual key of SEG2 to the DB2 primary key in the target DB2 tables TAB2 and TAB3.
5. If the conceptual fully concatenated key of a segment includes ID fields of a physical ancestor/parent, you must define the propagation request with the IMS DPROP PATH=ID option. Include as PATH data ID fields that are part of the conceptual key of the physical parent/ancestor. Also specify the PATH data in the EXIT= keyword of the IMS DBD.
6. ID fields must be defined in the IMS DBD. The IMS name in the IMS DBD and the IMS DPROP name in the propagation request definition of these ID fields must be the same. One exception to this rule is ID fields of internal segments. Usually, you cannot define the ID fields of internal segments in the IMS DBD.

For mapping case 2, an extension segment cannot participate in mapping to the DB2 primary key. Extension segments must not have an IMS key field or propagated dependent segments.

For mapping case 3, the entity segment is an internal segment, also called an embedded structure. As with other types of entity segments, internal segments with more than one occurrence must be uniquely identifiable. The internal segment must be identified through ID fields. If the internal segment does not contain ID fields, you can use a Segment exit routine to construct ID fields in the edited segment format.

You do not need to define ID fields of internal segments in the IMS DBD. And you can change the ID fields of internal segments during an IMS REPL. Changes are interpreted by IMS DPROP as a sequence of deletes and inserts of occurrences of the internal segment.

Key rule 4: Key Rule 4 describes how the DB2 foreign key of a dependent table should be mapped when defining DB2 RIRs that match IMS parent/child relationships.

When defining a matching DB2 RIR, the DB2 foreign key in the child table must be mapped from the *same fields* as the DB2 primary key of the parent table. Therefore:

- The foreign key of the child table used to implement a DB2 parent/child RIR matching a *physical* IMS parent/child relationship must be the propagated IMS fully concatenated key of the physical parent. If you are including ID fields, the foreign key must be the propagated conceptual fully concatenated key of the physical parent. So, the foreign key of the child table must be the propagated IMS physical concatenated key of the child segment or the propagated concatenated key of the child segment.
- The foreign key of the child table used to implement a DB2 parent/child RIR matching a *logical* IMS parent/child relationship must be the propagated IMS fully concatenated key of the logical parent. So, the foreign key of the child table must be the propagated IMS logical concatenated key of the child segment.

Key rule 5: If you choose to use CCU direct verification, observe the following rules:

1. Each propagated entity segment and each of its physical ancestors must have either a unique IMS key or only one occurrence under its parent, with the exception of internal segments.
2. The ordering sequence of the index for the DB2 primary key and the IMS fully concatenated key must be the same:
   - The root key must map to the highest order columns of the primary key index, the key from the dependent segment must map to the next highest order part of the primary key index, and so on.
   - In IMS, sequencing is done based on the hexadecimal values of the bytes in fields. Use the index of the DB2 primary key to maintain the sequencing.

     If you map signed numeric IMS fields that have both positive and negative values, you might lose your matched sequence.

     Sequencing will not match when propagation involves IMS HDAM databases and DEDBs without sequential randomizers. The ordering sequence of the index of the DB2 primary key and the IMS fully concatenated key are not the same.

You can use CCU's direct technique even though internal segments are not stored in a particular sequence within the containing segment.

For more information on the CCU and direct verification, see "Overview of the CCU" on page 203 and the*Reference*.

Key rule 6: We recommend for HIDAM, HISAM, SHISAM, and HDAM and DEDBs with sequential randomizers, the DB2 index for the DB2 primary key should be clustered. The ordering sequence of this index should be the same as the IMS fully concatenated key.

For HDAM and DEDBs without sequential randomizers, the DB2 table should be clustered in the same sequence as the physical HDAM/DEDB sequence, if practical.

Example of mapping keys in ideal case (PRTYPE=E)

Figure 12 on page 56 shows the ideal case for mapping keys with PRTYPE=E. All entity segments have a unique IMS fully concatenated key. The DB2 primary key is the propagated IMS fully concatenated key.

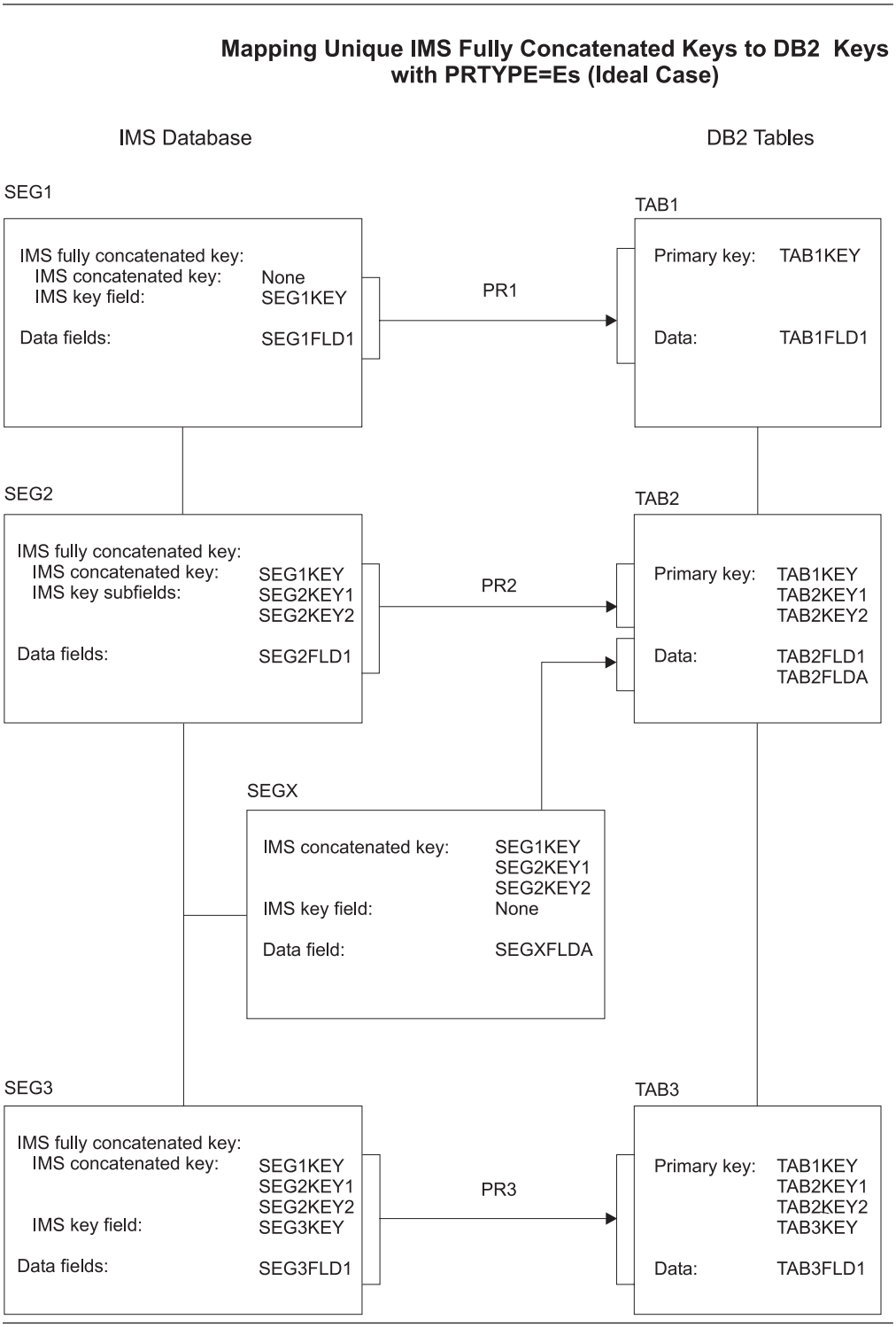


Figure 12. Mapping unique IMS fully concatenated keys to DB2 primary keys with PRTYPE=Es (Ideal case)

Propagation requests 1 and 3 use mapping case 1. Propagation request 2 uses mapping case 2.

- As required by key rule 1, all tables have a DB2 primary key.
- As required by key rule 2, every byte of the IMS fully concatenated key of each entity segment is propagated to the DB2 primary key. Propagation request 1

maps the entity segment's key as one field to a single column. Propagation request maps the entity segment's key as multiple subfields to multiple DB2 columns.

- As is the ideal case for key rule 3, each entity segment has a unique IMS key and a unique IMS fully concatenated key. The DB2 primary key is the propagated IMS fully concatenated key of the entity segment.
- DB2 RIRs have been implemented. As required by key rule 4, the DB2 foreign key of a child table is the propagated IMS concatenated key of the child segment.
- According to key rule 5, CCU's direct verification technique can be used for all segments since each propagated segment and each of its ancestors have a unique IMS key, assuming the ordering sequence of the index for the DB2 primary key and the ordering sequence of the IMS fully concatenated key are the same.

Example of mapping keys in non-ideal case (PRTYPE=E)

Figure 13 on page 58 illustrates another case for mapping keys with PRTYPE=E. In this case, some entity segments do not have unique IMS fully concatenated keys, but they do have a unique conceptual fully concatenated key. The primary DB2 key is the propagated conceptual fully concatenated key.

Mapping Unique Conceptual Fully Concatenated Keys to Primary DB2 Keys with PRTYPE=E (Non-Ideal Case)

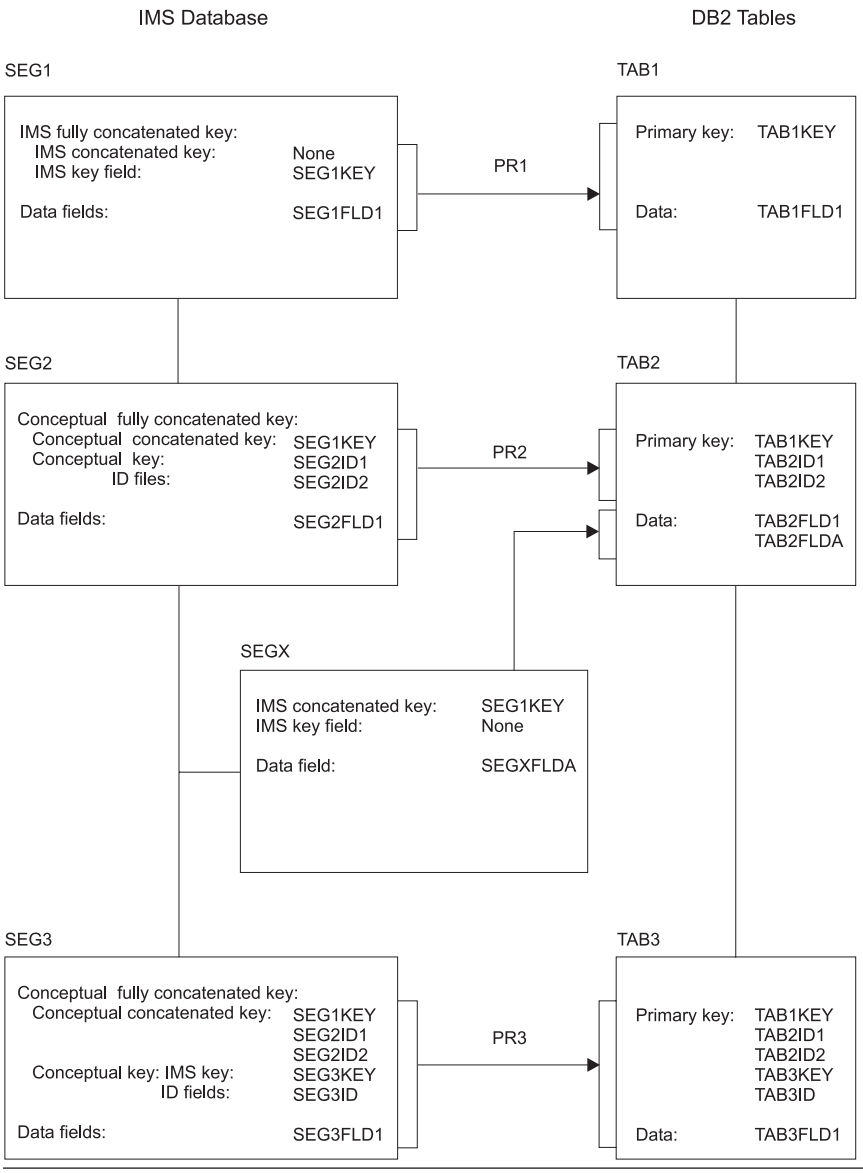


Figure 13. Mapping unique conceptual fully concatenated keys to primary DB2 keys with PRTYPE=E (Non-ideal case)

Figure 13 shows the key rules for PRTYPE=E when the entity segments SEG2 and SEG3 do not have unique IMS key fields. By combining ID fields with the IMS fully concatenated key, you can identify each entity segment with a unique conceptual fully concatenated key.

Propagation requests 1 and 3 use mapping case 1. Propagation request 2 uses mapping case 2.

- As required by key rule 1, all tables have a DB2 primary key.
- As required by key rule 2, every byte of the IMS fully concatenated key of each entity segment is propagated to the DB2 primary key.

- As required by key rule 3, you can identify each entity segment either through a unique IMS fully concatenated key or through a unique conceptual fully concatenated key. The DB2 primary key is either the propagated unique IMS fully concatenated key or the propagated unique conceptual fully concatenated key.
  - The entity segment SEG1 has a unique IMS key field and a unique IMS fully concatenated key: SEG1KEY. The DB2 primary key Tabbed of Tab is the propagated unique IMS fully concatenated key of Sage.
  - The entity segment SEG2 has no IMS key field. The ID fields SEG2ID1 and SEG2ID2 of segment SEG2 are used to uniquely identify SEG2. The conceptual key of SEG2 consists of SEG2ID1 and SEG2ID2. The conceptual fully concatenated key of SEG2 is the combination of SEG1KEY, SEG2ID1, and SEG2ID2; it is mapped to the DB2 primary key of TAB2.
  - The entity segment SEG3 has a non-unique IMS key field SEG3KEY. The combination of SEG3KEY and ID field SEG3ID are used to uniquely identify entity segment SEG3. The conceptual key of SEG3 consists of SEG3KEY and SEG3ID. The conceptual fully concatenated key of SEG3 is the combination of SEG1KEY, SEG2ID1, SEG2ID2, SEG3KEY, and SEG3ID; it is mapped to the DB2 primary key of TAB3.

  As required by key rule 3, the conceptual key of SEG2 is defined identically in PR2 and PR3 as being the combination of SEG2ID1 and SEG2ID2. Both PR2 and PR3 include the same ID fields in the conceptual key of SEG2; and both propagation requests map the same ID fields of SEG2 to the DB2 primary key of TAB2 and TAB3.
- DB2 RIRs have been implemented. As required by key rule 4, the DB2 foreign key of a child table is the propagated conceptual concatenated key of the child segment.
- According to key rule 5, CCU's direct verification cannot be used for checking the consistency of SEG2 and SEG3 with TAB2 and TAB3, because SEG2 and SEG3 have neither a unique IMS key field nor a maximum of one occurrence under their parent. You can, however, use CCU's hashing technique.

Rules For PRTYPE=L (Limited Function)

Figure 14 on page 62 shows the key mapping rules of PRTYPE=Ls.

Key Rule 1: As with PRTYPE=E and L, a propagated DB2 table must have a primary key. All columns of the DB2 primary key must be mapped by the propagation request. The IMS fields that map to the DB2 primary key must result in a unique DB2 primary key.

Key Rule 2: Not applicable.

Key Rule 3: Ideally, a propagated entity segment has a *unique IMS fully concatenated key*, meaning the entity segments and their physical parent/ancestors have either a unique IMS key field or a maximum of one occurrence under their physical parents.

Also, the DB2 primary key is the propagated IMS fully concatenated key of the entity segment, meaning the DB2 primary key is mapped by the IMS fully concatenated key of the entity segment; and the IMS fully concatenated key of the entity segment is mapped to the DB2 primary key. The fields being mapped to the DB2 primary key must not overlap.

If a propagated entity segment does not have a unique IMS fully concatenated key, then create a unique ID by combining fields. Combine:

- Fields located in its IMS fully concatenated key
- Fields located in the data portion of the entity segment, the data portion of its physical parent and physical ancestors, or both

When in the mapping to the DB2 primary key and including fields in the data portion of the entity segment or physical parent/ancestor, observe the following rules:

1. The field values can change, unless the results of change violates optional DB2 RIRs or the fields are PATH data fields of a propagation request defined with PATH=ID. If the field values cannot change, propagation fails.
2. IMS application programs should not insert segment occurrences that violate the uniqueness rule of the target DB2 primary key or propagation fails.
3. For variable-length segments, fields mapped to the DB2 primary key must be located in the existing portion of the variable-length segment, or propagation fails.
4. When in the mapping to the DB2 primary key and including data fields of a physical ancestor/parent, define the propagation requests with the PATH=ID or DENORM option. Also define the PATH data in the EXIT= keyword of the IMS DBD.
5. Non-key IMS fields that are mapped to the DB2 primary key must be defined in the IMS DBD. The IMS name in the IMS DBD and the IMS DPROP name in the propagation request definition of these ID fields must be the same. One exception to this rule is ID fields of internal segments. Usually, you cannot define the ID fields of internal segments in the IMS DBD.

For mapping case 2, an extension segment cannot participate in mapping to the DB2 primary key. For mapping case 3, the entity segment is an internal segment, also called an embedded structure. As with other types of entity segments, internal segments with more than one occurrence must be uniquely identifiable. The internal segment must be identified through ID fields. If the internal segment does not contain ID fields, you can use a Segment exit routine to construct ID fields in the edited segment format.

You can change the ID fields of internal segments during an IMS REPL. Changes are interpreted by IIMS DPROP as a sequence of deletes and inserts of occurrences of the internal segment.

Key Rule 4: Key Rule 4 describes how the DB2 foreign key of a dependent table should be mapped when defining DB2 RIRs that match IMS parent/child relationships.

The definition of matching DB2 RIRs is optional. If you implement RIRs, use them as a subset of IMS parent/child relationships. See “DB2 referential integrity guidelines” on page 45 for more information on this subject.

When defining a matching DB2 RIR, the DB2 foreign key in the child table must be mapped from the *same fields* as the DB2 primary key of the parent table. Therefore:

- The foreign key of the child table used to implement a DB2 parent/child RIR matching a *physical* IMS parent/child relationship is propagated from the same combination of fields as the DB2 primary key of the parent table. Combine:
  - Fields located in the IMS fully concatenated key of the physical parent segment
  - PATH data fields located in the data portion of the physical parent segment, physical ancestors

PATH data fields included in a foreign key should not change their value.

- The foreign key of the child table used to implement a DB2 parent/child RIR matching a *logical* IMS parent/child relationship is propagated from the same combination of fields in the IMS fully concatenated key of the logical parent segment as the primary key of the parent table.

Key rule 5: Same as PRTYPE=E and L. See page 55.

Key rule 6: Same as PRTYPE=E and L. See page 55.

Example of mapping keys (PRTYPE=L)

Figure 14 on page 62 shows mapping keys with PRTYPE=L. In the example, the entity segment SEG2 cannot be identified uniquely by combining the IMS fully concatenated key and ID fields that do not change their value. SEG2 has, therefore, no unique conceptual fully concatenated key and cannot be propagated by using PRTYPE=E.

Propagation requests 1 and 3 use mapping case 1. Propagation request 2 uses mapping case 2.

- As required by key rule 1, all tables have a DB2 primary key.
- As required by key rule 3:
  - Entity segment SEG1 has a unique IMS fully concatenated key, SEG1KEY. The DB2 primary key of TAB1 is the propagated IMS fully concatenated key.
  - Entity segment SEG2 is uniquely identifiable, even though it cannot be identified by combining the IMS fully concatenated key and ID fields. SEG2 is uniquely identifiable by combining its IMS fully concatenated key SEG1KEY with its data fields SEG2FLD1 and SEG2FLD2.

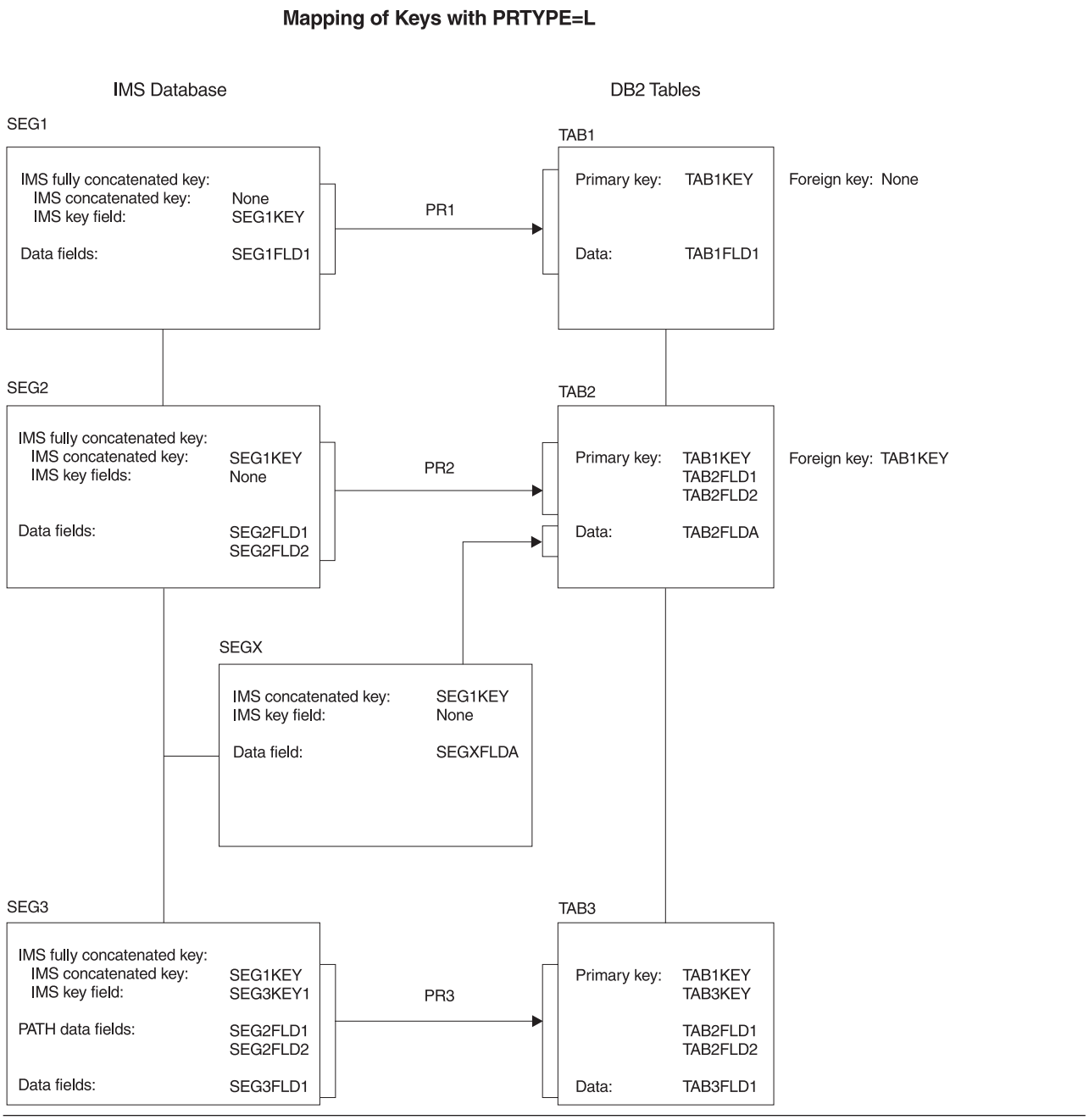


Figure 14. Mapping of keys with PRTYPE=L

Map to the DB2 primary key of TAB2 by combining:
- Fields located in the IMS fully concatenated key
- Data fields SEG2FLD1 and SEG2FLD2

In Figure 14, fields SEG2FLD1 and SEG2FLD2, used to uniquely identify SEG2, can change their values.

– You can also make SEG3 uniquely identified by combining its IMS fully concatenated key (SEG1KEY and SEG3KEY) with the data fields SEG2FLD1, SEG2FLD2, and SEG3FLD1. SEG2FLD1 and SEG2FLD2 are PATH data fields located in a physical ancestor. SEG3FLD1 is located in the entity segment.

Map to the DB2 primary key of TAB3 by combining fields in the IMS fully concatenated key, the data portion of the entity segment, and the data portion of physical ancestors.

- DB2 RIRs have only been implemented between TAB1 and TAB2.

  As required by key rule 4, the foreign key in the child table TAB2 is mapped from the same field as the DB2 primary key of the parent table TAB1: SEG1KEY.

Assuming SEG2 has neither a unique IMS fully concatenated key nor ID fields that allow a conceptual key to be built, the following conditions apply:

- PR2 cannot be defined as PRTYPE=E, because key rule 3 for PRTYPE=E requires that the entity segment have a unique conceptual fully concatenated key. PR3 cannot be defined as PRTYPE=E.
- Matching DB2 RIRs cannot be implemented between TAB2 and TAB3. Matching DB2 RIRs require that the foreign key in the child table TAB3 be mapped from the same fields as the primary key of TAB2, SEG1KEY and PATH data fields SEG2FLD1 and SEG2FLD2. Implementing matching RIRs would also require that SEG2FLD1 and SEG2FLD2 not change their value.

Comparison of key mapping rules by propagation request type

Table 5. Comparison of key mapping rules by propagation request type

PRTYPE=E	PRTYPE=L
Key rule 1	
A propagated DB2 table must have a DB2 primary key. All columns of the DB2 primary key must be mapped by the propagation request.	Same as for PRTYPE=E.
Key rule 2	
All of the IMS fully concatenated key of an entity segment must be propagated to the DB2 primary key. For a propagated logical child segment, the entire logical concatenated key must also be propagated.	N/A
Key rule 3	
Ideally for each propagated entity segment: • The propagated entity segment should have a unique IMS fully concatenated key. • The DB2 primary key should be the propagated IMS fully concatenated key of the entity segment.	Same as PRTYPE=E in ideal situations.
If a propagated entity segment does not have a unique IMS fully concatenated key, then:	Then:
• The entity segment must have a unique conceptual fully concatenated key.	• The entity segment must be uniquely identifiable, by combining: – Fields located in its IMS fully concatenated key – Fields located in the data portion of the segment or in the data portion of its physical parent/ancestors
• The DB2 primary key must be the propagated conceptual fully concatenated key of the entity segment.	• The DB2 primary key must be mapped from: – Fields located in its IMS fully concatenated key – Fields located in the data portion of the segment or in the data portion of its physical parent/ancestors You do not need to completely map the IMS fully concatenated key to the DB2 primary key.

Table 5. Comparison of key mapping rules by propagation request type (continued)

PRTYPE=E	PRTYPE=L
• Fields contributing to the conceptual fully concatenated key must not change their values.	• Fields being mapped to the DB2 primary key can change their value, unless the results of change violate optional DB2 RIRs or the fields are PATH data fields of a propagation request defined with PATH=ID.
Extension segments of mapping case 2 propagation requests must not have an IMS key field, participate in mapping to the DB2 primary key, nor have dependent segments propagated by PRTYPE=E.	Extension segments of mapping case 2 propagation requests must not participate in mapping to the DB2 primary key.
Key rule 4	
The foreign key in the child table must be mapped from the same fields as the primary key of the parent table.	
The foreign key used to implement a DB2 parent/child RIR matching a *physical* IMS parent/child relationship:	
• Must be the propagated IMS/conceptual fully concatenated key of the physical parent segment.	• Is propagated from the same combination of: – Fields in the IMS fully concatenated key of the physical parent segment – PATH data fields in the data portion of the physical parent segment and/or physical ancestors as the DB2 primary key of the parent table. PATH data fields included in a foreign key should not change their value.
The foreign key used to implement a DB2 parent/child RIR matching a *logical* IMS parent/child relationship:	
• Must be the propagated IMS fully concatenated key of the logical parent segment.	• Is propagated from the same combination of fields in the IMS fully concatenated key of the logical parent segment as the DB2 primary key of the parent table.
Key rule 5	
See description of key rule 5 on page 55.	
Key rule 6 (A recommendation)	
For HIDAM, HISAM, SHISAM, and HDAM and DEDBs with sequential randomizers, the DB2 index for the DB2 primary key should be clustered. The ordering sequence of this index should be the same as the IMS fully concatenated key. For HDAM and DEDBs without sequential randomizers, the DB2 table should be clustered in the same sequence as the physical HDAM/DEDB sequence, if practical.	

Supported field formats and conversions

IMS DPROP supports a number of field formats and field format conversions, as shown in Table 6 on page 66. You can implement additional field formats and conversions using Field exit routines. For more information on Field exit routines, see the *Customization Guide*.

You do not need to propagate every data field to or from the target DB2 tables. And you can map a field to multiple columns in the same table. For more information on this subject, see “Mapping between fields and columns” on page 43.

You can expand or reduce fields and columns when they are propagated. The DB2 table column does not need to have the same format as its IMS counterpart. However, be careful that the new format meets the needs of the data. Generally, it is easier to maintain the same format on all copies of the data.

Topics described in this section include:
- Describing fields
- Converting data
- A summary of conversion rules
- Characteristics of supported IMS data types
- Mapping and conversion between numeric fields
- Mapping and conversion between non-numeric data

Describing fields

To map IMS fields to or from DB2 columns, you must describe each field to be propagated. You do not need to describe DB2 columns, because IMS DPROP gets column descriptions from the DB2 catalog. When describing an IMS field, provide the following information:

- A field name
- The starting position of the field within the IMS segment
- The type of data format (such as small integer, fixed-length character)
- The length of the field, for those data types that do not have an inherent length; for example, a small integer has an inherent length of two bytes
- The scale of the field, for decimal packed and decimal zoned fields

Provide the field description as part of the propagation request definition. Describe the fields to DataRefresher or in the MVG input tables. You do not have to define each propagated field in the IMS DBD; IMS DPROP ignores DBD field definitions (except DBD field definitions for key fields and fields mapped to the primary DB2 key).

Describe the fields as they appear in the I/O area of an IMS call that accesses the IMS segment through a PCB. Do not include field sensitivity but do reference a physical DBD.

If you use a Segment exit routine, describe the fields as they appear in the segment format that has been defined to IMS DPROP. (During IMS-to-DB2 mapping, the field you describe is the segment *after* it has been processed by the exit.) IMS DPROP does not understand field formats in the IMS database format of the segment.

For fields that are processed by a Field exit routine, describe the field before and after its processing by the Field exit routine.

IMS DPROP's generalized mapping cases require that each field have a fixed starting position within each segment. When segments you want to propagate do not have a fixed starting position, you can use Segment exit routines to create a fixed starting position for each field.

Converting data

IMS DPROP does data conversion during:
- Data propagation
- Execution of the CCU
- Execution of the DLU

IMS DPROP supports:

- All IMS field formats supported by DXT 2.5
- All column formats supported by DB2 2.2

- Field format conversions done during an extract and load process using DataRefresher with the DB2 Load utility, except for:
  - Binary integer and decimal number to floating point
  - Floating point to binary integer and decimal number
  - Timestamp to date/time
  - Date/time fields in formats other than ISO, USA, EUR, or JIS

IMS DPROP can automatically convert data. A summary of all data formats and conversions supported by IMS DPROP is shown in Table 6.

Table 6. Data conversions supported by IMS DPROP.

"R" means a recommended pair of IMS field and DB2 column definitions.

"S" means supported pair of IMS field and DB2 column definitions.

DataRefresher and IMS DPROP Definitions of the IMS Fields	DB2 Column Definitions													
	SMALLINT	INTEGER	DECIMAL	FLOAT(21)	FLOAT(53)	CHAR	VARCHAR	LONG VARCHAR	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DATE	TIME	TIMESTAMP
B (SINGLE BYTE BINARY)	S	S	S											
H (SMALLINT)	R	S	S											
F (INTEGER)	S	R	S											
P (PACKED)	S	S	R											
Z (ZONED)	S	S	R											
E (SINGLE FLOAT)				R	S									
D (DOUBLE FLOAT)				S	R									
C (CHAR)						R	S	S						
VC (VARCHAR)						S	R	R						
G (GRAPHIC)									R	S	S			
VG (VARGRAPHIC)									S	R	R			
A (DATE)												R		
T (TIME)													R	
S (TIMESTAMP)														R

When possible, the format of the fields defined in the DB2 tables should match the format of the fields defined to IMS. If they do not and the supported conversions do not satisfy your requirements, you must write a Field exit routine.

Summary of conversion rules

During the mapping of numeric fields and columns, the whole part of a decimal or integer number is never truncated. Conversions requiring truncation of the whole part cause propagation to fail. If necessary, leading zeros are added or deleted to the integer part of the numeric field. IMS DPROP might also truncate or drop the

fractional part, if necessary. Truncation or dropping of numbers is not considered an error, and no warning message is issued. Trailing zeros are appended to the fractional part of a decimal number as needed.

When the target of mapping is a decimal number, a decimal number with the appropriate sign is produced (hexadecimal C or X'C' for a positive number, or X'D' for a negative number). If necessary, you can use a Field exit routine to produce, decimal signs other than X'C' and X'D' during DB2-to-IMS mapping.

During the mapping of character strings, if the source is longer than the target, even after IMS DPROP has eliminated trailing blanks, propagation fails. If the source is shorter than the target, trailing blanks are added.

For the mapping of graphic strings, the rules that apply are similar to those for character strings. If the source is longer than the target after eliminating trailing double character blanks, propagation fails. If the source is shorter than the fixed-length target, then trailing double character blanks are appended.

Characteristics of supported IMS data types

This section describes the types of IMS data supported by IMS DPROP.

Single-byte binary field

A single-byte binary field is an unsigned binary integer with a precision of eight bits. The content of a single-byte binary field is assumed to be a positive number, unlike fields with other numeric data types, which can contain both positive and negative values. The field can have a value between 0 and 255.

Small integer field

A small integer field is a signed binary integer with a precision of fifteen bits. Small integers are sometimes referred to as *halfword* binary integers.

Large integer field

A large integer field is a signed binary integer with a precision of thirty-one bits. Large integers are sometimes referred to as *fullword* binary integers.

Decimal packed field

A decimal packed field is a packed decimal number with an implicit decimal point. You describe the position of the implied decimal point by specifying a scale attribute.

The length of a decimal packed field is 1 to 16 bytes. Therefore, a decimal packed field can have 1 to 31 digits.

A DB2 column with decimal packed format always has X'C' or X'D' for its sign.X'C' is the positive sign, and X'D' is the negative sign. Packed fields in an IMS database may have positive signs of X'A', X'C', andX'F', and negative signs of X'B' and X'D', depending on how the installation's applications have been designed.

Decimal zoned field

A decimal zoned field is a decimal number with an implicit decimal point. Decimal zoned fields are also sometimes called *unpacked* decimal fields. Describe the position of the implied decimal point by specifying a scale attribute. The length of a decimal zoned field is 1 to 16 bytes.

Zoned fields in an IMS database may have the positive signs of X'A', X'C', and X'F', and negative signs of X'B' and X'D' (depending on how the installation's applications have been designed).

Single-precision floating point number
A single-precision floating point number is a short (32 bit) floating-point number. This is what DB2 calls FLOAT(21).

IMS DPROP does not support floating point numbers in the WHERE clause of a propagation request.

Double-precision floating point number
A double-precision floating point number is a long (64 bit) floating-point number. This is what DB2 calls FLOAT(53).

IMS DPROP does not support floating point numbers in the WHERE clause of a propagation request.

Fixed-length character field
A fixed-length character field is a string of bytes having a fixed length. The length of this field is 1 to 254 bytes.

Variable-length character field
A variable-length character field is a string of bytes having a variable length.

For variable-length IMS fields, IMS DPROP requires that the field length be stored in a separate field. This separate field can have any numeric data type except floating point. Its scale must be zero, and it must be located before the variable-length field.

Variable-length fields returned by Field exit routines do not have a length field associated with them. The length must be returned by the exit routine.

The defined maximum length of a variable-length character field is 1 to 32,767 bytes.

Fixed-length graphic field
A fixed-length graphic field is a sequence of double-byte characters having a fixed length. The length of such a field is 2 to 254 bytes (1 to 127 DBCS characters).

Variable-length graphic field
A variable-length graphic field is a sequence of double-byte characters having a variable length.

For variable-length IMS fields, IMS DPROP requires that the length of the field be stored in a separate field. This separate field can have any numeric data type except floating point. The length should be expressed in bytes, not DBCS characters. Its scale must be zero, and it must be located before the variable-length field.

Variable-length fields returned by Field exit routines do not have a length field associated with them. The length must be returned by the exit routine.

The defined maximum length of a variable-length graphic field is 2 to 32,766 bytes (1 to 16,383 DBCS characters).

Date Date is a three-part value: year, month, and day. It has a length of 10 bytes. DB2, DataRefresher, and IMS DPROP support dates in the following type of formats: ISO, USA, EUR, JIS, and LOCAL (LOCAL is site-defined).

For DB2 columns, IMS DPROP supports all of the preceding date formats without a Field exit routine.

For IMS fields, IMS DPROP supports all of the preceding date formats with the exception of LOCAL. LOCAL requires use of a Field exit routine.

Unless you use a Field exit routine, mapping for DB2-to-IMS synchronous propagation defaults to the DATE format you specified during IMS DPROP system generation.

DataRefresher users should use Field exit routines (DataRefresher calls them Data Type exits) instead of DataRefresher Date/Time Conversion User exits to convert date fields so that you can use the same exit routine for both DataRefresher and IMS DPROP.

Time Time is a three-part value: hour, minute, and second. It has a length of 8 bytes. DB2, DataRefresher, and IMS DPROP support times in the following type of formats: ISO, USA, EUR, JIS, and LOCAL (LOCAL is site-defined).

For DB2 columns, IMS DPROP supports all of the above time formats without use of a Field exit routine.

For IMS fields, IMS DPROP supports all of the preceding time formats with the exception of LOCAL. The LOCAL time format requires the use of a Field exit routine.

Unless you use a Field exit routine, mapping for DB2-to-IMS synchronous propagation defaults to the TIME format you specified during IMS DPROP system generation.

DataRefresher users should use Field exit routines (DataRefresher calls them Data Type exits) instead of DataRefresher Date/Time Conversion User exits to convert time fields so that you can use the same exit routine for both DataRefresher and IMS DPROP.

Timestamp

A timestamp is a seven-part value, containing year, month, day, hour, minute, second, and microsecond. The length of a timestamp field is 19 to 26 bytes.

The complete string representation of a timestamp is: *yyyy.mm.dd.hh.mm.ss.nnnnnn*

You can truncate or omit microseconds. You can omit leading zeroes from the month, day, and hours.

Mapping and conversion between numeric fields

IMS DPROP does conversion between numeric data types as follows:

- During the mapping of numeric fields, the whole part of a decimal or integer number is not truncated. Conversions that require truncation of the whole part cause propagation to fail. If necessary, leading zeroes are appended to or eliminated from the whole part.
- During mapping of numeric fields, IMS DPROP might truncate or drop the fractional part. Truncation or dropping of numbers is not considered an error, and no warning message is issued. Do not let the fractional part of a field or column used in a key be truncated.

  Trailing zeros are appended to the fractional part of a decimal number as needed.
- Mapping to a decimal number results in a number with the appropriate sign (X'C' for a positive number and X'D' for a negative number). If necessary, you can use a Field exit routine to produce, decimal signs other than X'C' and X'D' during DB2-to-IMS mapping.

  Be careful not to use decimal fields with signs other than X'C' and X'D' as IMS keys. See “Mapping and conversion between decimal fields” on page 70 for more details.

- Mapping from a single-precision floating point number to a double-precision floating point number is done by padding the single-precision data with eight hexadecimal zeroes.
- Mapping from a double-precision floating point number to a single-precision floating point number is done by converting and rounding the double-precision data to the single-precision format.

Mapping and conversion between binary integers

Mapping between two small integers or two large integers is straightforward. However, mapping and conversion between integers is more difficult when the format of the source and target are different. Propagation can fail if the target is not large enough to contain the source data.

If a binary integer field is part of an IMS key field and can have a negative value, the key sequence of IMS segments and DB2 columns will probably be different and prevents use of CCU's direct technique. Information on the direct technique is in "CCU verification techniques" on page 207.

Mapping and conversion between decimal fields

Mapping and conversion between decimal fields requires careful planning if the precision and scale of source and target are not identical.

- Propagation can fail if the whole part of the target is not large enough to contain the whole part of the source data.
- Truncation of the fractional part can result in problems in two-way propagation environments. You might lose fractional information in both the field that has the smaller scale *and* the field that has the larger scale.
- Truncation of the fractional part of a decimal *key* field can cause uniqueness of the key to be lost. For example, when the last digit of the fractional part of the two key values 123.45 and 123.46 is truncated, they are mapped to the same key field value, 123.4, which is not unique.

For decimal key fields and ID fields, you might encounter other problems because IMS and DB2 handle signed fields differently.

- In IMS, the same numerical decimal value with different positive signs X'A', X'C', and X'F' is considered to have different values. DB2 considers the values identical. For example, IMS considers the two packed fieldsX'123C' and X'123F' to have two different values. Therefore, you can have two different IMS segments with the *unique* key values of X'123C' and X'123F'. The mapping of these two packed values into a DB2 decimal column results in the same DB2 decimal value X'123C'. IMS also considers the negative signs X'D' and X'B' different, while DB2 considers them identical.

  Results can be unpredictable if an IMS decimal key field has multiple different positive or negative signs. For example, the propagation of a successful IMS insert of a root segment with an IMS key X'123C' fails because DB2 considers X'123C' a duplicate if the IMS database also contains a root with the IMS key X'123F'.

If the decimal field contains negative values, the key sequence of IMS segments and DB2 columns is different. You cannot use CCU's direct technique.

Recommendations: Define propagated decimal DB2 columns with the same precision and scale attributes as the corresponding decimal packed or zoned IMS fields to avoid:

- Potential propagation failures when the target field is not large enough to contain the whole part of the source data
- Loss of uniqueness when truncation occurs

Also, examine the signs used for decimal IMS keys and determine the potential for impact if keys have signs other than X'C' and X'D'.

Mapping and conversion between binary integers and decimal fields

Mapping between binary integers and decimal numbers is supported but not recommended. You should define the format of propagated DB2 columns to avoid mapping between binary integer and decimal fields.

Problems with mapping between decimal and binary integers include:

- Propagation can fail when the whole part of the target field is not large enough to contain the whole part of the source data.
- Loss of the fractional part of a decimal key can cause the uniqueness of the key to be lost.

For decimal IMS keys and ID fields, you might encounter additional problems:

- Fields with the same numerical decimal value but different positive signs (X'A', X'C', and X'F') are considered different; IMS also considers the negative signs X'D' and X'B' different. When mapping an IMS decimal key with different positive or negative signs to an integer DB2 column, you cannot preserve the difference in the IMS signs.

  Results can be unpredictable if an IMS decimal key has multiple different positive or negative signs.

If keys contain negative values, the key sequence of IMS segments and DB2 columns is different. You cannot use CCU's direct technique.

Mapping and conversion between floating point numbers

Mapping between two single-precision and two double-precision floating point numbers creates problems for IMS keys because the same floating point number is represented with different bit combinations in IMS and DB2.

Avoid conversion from a double- to a single-precision floating point number for keys, because it can cause loss of uniqueness and unpredictable results.

Recommendations: To avoid unpredictable results, do not use floating point numbers as keys. Also, define propagated DB2 columns so that conversion between single- and double-precision floating point numbers is avoided.

Mapping and conversion between non-numeric data

The following sections describe how IMS DPROP does conversion and mapping between character, graphic (DBCS), and date/time fields.

Mapping and conversion between character/graphic strings

IMS DPROP uses the following logic for conversions between character strings:

- When the source is longer than the target, trailing blanks are eliminated. Afterwards, propagation fails if the source is still longer than the target.
- If the source is shorter than the fixed-length target, IMS DPROP appends trailing blanks (for character strings) or double character blanks (for graphic strings) to the source data.

- If the source is smaller or equal to the maximum length of a variable-length target, the length of the target will be equal to the length of the source data.

IMS DPROP uses the following logic to map a variable-length character/graphic field of zero length to the target DB2 column:
- If the target DB2 column is fixed length, it will contain blanks
- If the target DB2 column is variable length, it will have a length of zero

Unless the length field is beyond the current end of a variable-length segment occurrence, IMS DPROP does *not* map a variable-length character/graphic field to a DB2 null value.

Recommendations: Define propagated DB2 columns so that the IMS field and the corresponding DB2 column are both fixed-length or variable-length. The fixed length (or maximum length for variable-length fields) should be the same.

Mapping and conversion between dates

Support for a LOCAL date format in IMS fields requires use of a Field exit routine. Unless you use a Field exit routine, mapping for DB2-to-IMS synchronous propagation is done for IMS fields in the DATE format you specified during IMS DPROP generation.

Mapping and conversion between times

As described on page 68, support for a LOCAL time format in IMS fields requires use of a Field exit routine. Unless you use a Field exit routine, mapping for DB2-to-IMS synchronous propagation is done for IMS fields in the DATE format you specified during IMS DPROP generation.

Mapping and conversion between timestamps

IMS DPROP supports mapping from a 19- to 26-byte time stamp field in an IMS database to a DB2 TIMESTAMP column. If the IMS field has fewer than 26 bytes, IMS DPROP assumes the trailing digits of the microsecond portion are missing and provides zeroes in the missing microsecond portion.

Normalizing data

Normalizing is the process of reducing data relationships to a simpler form. You can use mapping case 2, mapping case 3, and the WHERE clause to normalize, in the DB2 copy, IMS data that is not well normalized.

When you use the PATH data option and combine non-key data from a parent and a child segment into one target DB2 table, you usually denormalize IMS data.

Avoid denormalizing data if you intend to use the DB2 copy of the data for operational applications. However, you can denormalize data to improve performance if you use the DB2 copy in read-only mode for queries in decision-support applications. See “PATH=DENORM: Denormalizing data to improve performance of DB2 queries” on page 28.

Refer to the *DB2 Administration Guide* for more information on normalization.

Chapter 4. Control information and environment

In addition to mapping your data, as described in Chapter 2, “Decisions affecting mapping and propagation,” on page 11 and Chapter 3, “Propagation guidelines, rules, and restrictions,” on page 39, you must prepare your operating environment.

This chapter describes:
- IMS DPROP control Information and environment control information
- IMS DPROP global master timestamp (Sysplex)
- Use of MVG input tables and the audit trail table
- Operational environments in which IMS DPROP runs

and gives guidance on how to prepare your environment for propagation.

IMS DPROP control information

This section discusses the MQ-DPROP control information located in the IMS DPROP directory, and VLF objects. Topics include:
- IMS DPROP directory
- Propagation status file
- IMS DPROP’s use of VLF

IMS DPROP directory

The IMS DPROP directory consists of multiple relational tables, created during IMS DPROP generation. Figure 15 on page 75 summarizes the content of the IMS DPROP directory, including the tables and the RIRs between tables. IMS DPROP tables are:

- The master table, which contains all required IMS DPROP system information.
- The following mapping tables:

Table	Description
PR	Contains one row for each propagation request generated. Each row contains all required information for the propagation request.
SEG	Contains one row for each segment type associated with each propagation request.
TAB	Contains a row for each DB2 table associated with each propagation request. For: – Generalized mapping cases, there is only one row – User mapping cases, there can be one or more rows
FLD	Contains one row for each IMS field defined in each propagation request. Each row describes the IMS field and the DB2 column to or from which it is to be propagated.
WHR	Contains one or more rows for each propagation request that has a WHERE clause associated with it, depending on the size of the clause.
MSG	Can contain zero, one, or more rows. Each row contains a warning or error message issued during the creation or the latest revalidation of each propagation request.

- The following control block tables:

CB Table	Description

RUP	Contains one RUP Propagation Request control block (PRCB) for each propagated segment type.
Receiver control table (RCT)	For IMS DPROP asynchronous propagation, contains a row for each Receiver created. It maintains the current status of each Receiver and the status of the most recently processed PRDS for each Receiver. If you implement synchronous or user asynchronous propagation, the table is not created.
PR control table (PRCT)	For each LOG-ASYNC propagation Receiver, this table contains a row for each propagation request assigned to that Receiver. It defines which propagation requests are assigned to which Receivers. If you implement LOG-ASYNC or user asynchronous propagation, the table is not created.
PRDS register table	For LOG-ASYNC propagation, this table contains a row for each registered PRDS. If you implement LOG-ASYNC or user asynchronous propagation, the table is not created.
PRDS volume table	For LOG-ASYNC propagation, this table is an extension of the PRDS register table and contains a row for each uncataloged PRDS in the PRDS register table. If you implement LOG-ASYNC or user asynchronous propagation, the table is not created.

The IMS DPROP directory tables are described in detail in the *Reference*.

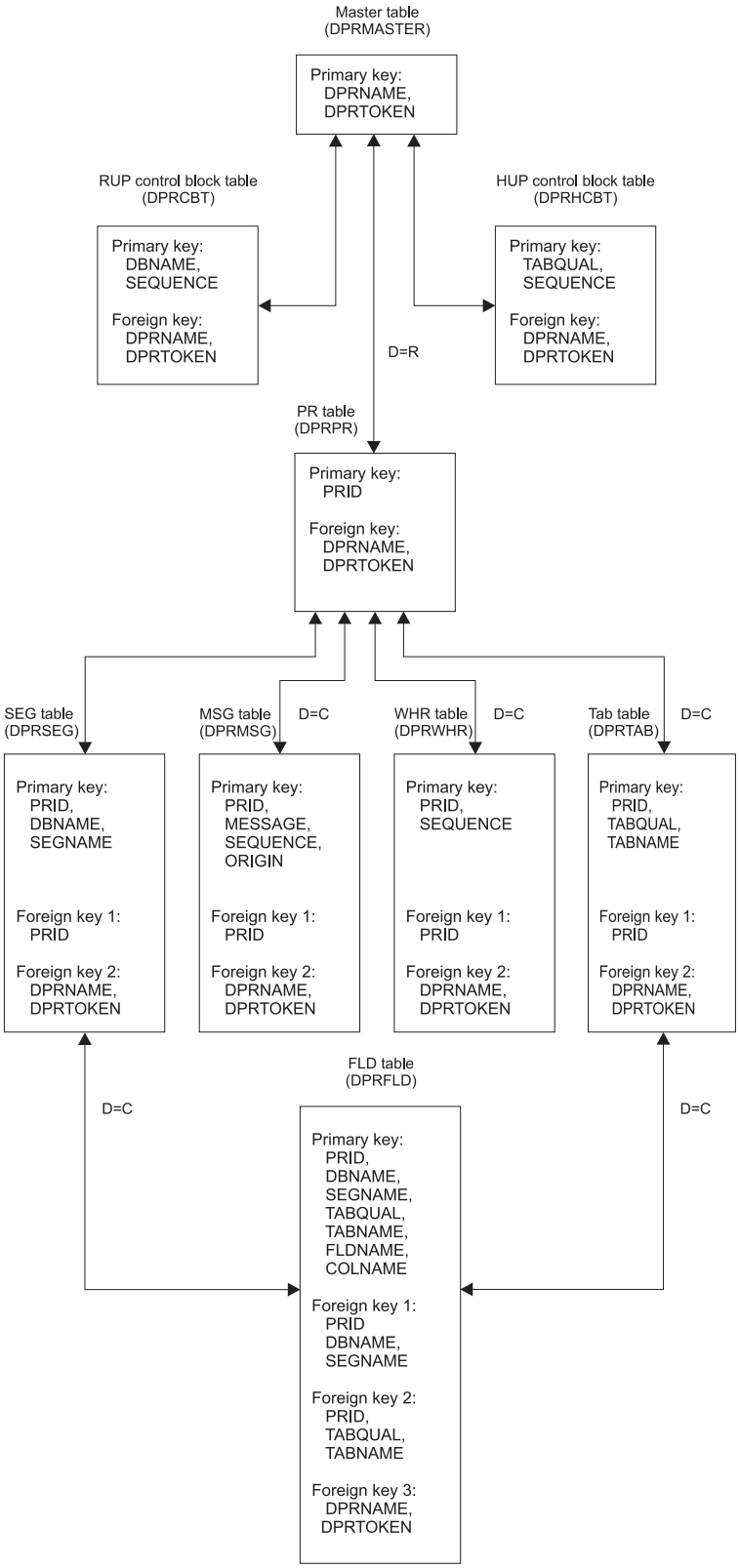


Figure 15. IMS DPROP directory

The control block tables cannot be used for queries because they contain mapping information in the internal control block format of RUP and HUP.

With the exception of the four IMS DPROP LOG-ASYNC Propagation only tables:
- RCT
- PRCT
- PRDS register table
- PRDS volume table

Each row of the IMS DPROP directory tables contains the directory ID.

The directory tables must be updated using IMS DPROP-provided programs initialized through the MVG and SCU. You must not update mapping tables with your own applications or QMF. You could cause inconsistencies between different mapping tables, or between the mapping tables and the control block tables or VLF objects. Inconsistencies can cause unpredictable propagation results and propagation failure. It is recommended that you limit the number of users with DB2 ALTER/DELETE/INSERT/UPDATE privileges for the IMS DPROP directory tables or database privileges such as IMS DPROP, RECOVERDB.

The MVG input tables are not considered part of the IMS DPROP directory; therefore, you can update them with your own applications or QMF.

Use of Vitrual Lookaside Facility (VLF)

IMS DPROP uses the MVS/ESA Virtual Lookaside Facility (VLF) to retrieve IMS DPROP control blocks from a data space. Using VLF reduces the path length required by propagation and reduces the possibility of DB2 enqueue conflicts between the RUP, HUP (synchronous only), and IMS DPROP utilities.

VLF Requirements

IMS DPROP creates and uses:

- One VLF object for each propagated segment type (each RUP PRCB).
- One VLF object for the master table (DPRMASTER). This consists of a single row containing the IMS DPROP system identifier and a master timestamp indicating the last time the IMS DPROP directory was changed.
- VLF objects to maintain counts of propagation failures:
  - One VLF object for each propagation request
  - One VLF object for each IMS DPROP system
- One PDS VLF object for each global master timestamp (GMTS) (Sysplex)

For information about how to provide SYS1.PARMLIB specifications to enable IMS DPROP to use VLF, refer to *Installation*. For information on VLF, refer to *OS/390 MVS Application Development Guide* and *OS/390 MVS Initialization and Tuning*.

Initializing, refreshing or recreating VLF objects

You can initialize and refresh the contents of the VLF class used by your IMS DPROP system using the SCU INIT VLF control statement. Use this statement after each IPL to initially populate VLF with the appropriate IMS DPROP objects. The SCU INIT VLF control statement reduces the probability of DB2 enqueue conflicts for the IMS DPROP directory tables during propagation.

When user or operator errors create inconsistency between VLF objects and their IMS DPROP directory counterparts, you can use the SCU INIT VLF control statement to refresh the VLF objects from the IMS DPROP directory.

To recreate the PRCB table and VLF objects from the directory tables, use the RECREATE control statement of the MVGU.

Use of the Global Master Timestamp (GMTS) for Sysplex

IMS DPROP uses a Global Master Timestamp (GMTS) to signal changes in control information to the various IMS DPROP components on multiple MVS systems. IMS DPROP implements the GMTS as a PDS member in a VLF PDS-class of objects to increase performance of its components.

In a Sysplex environment, when an object is added, updated or deleted in a VLF space, the other VLF spaces are notified that their copy of that VLF object is no longer valid. Notification is available only for *PDS-classes* of VLF objects. VLF PDS-classes are classes of objects created from members of a partitioned data set (PDS).

You can use a security product, such as RACF to ensure all updates to the VLF copy of the GMTS are made in authorized mode through the IMS DPROP supervisor call routine (SVC).

For more information on Sysplex systems and how to set them up see*MVS/ESA Setting up a Sysplex Document Number GC28-1449-00* and*System/390 MVS Sysplex Overview GC28-1208-00*.

How GMTS works

The GMTS is stored as a single record in a member of a PDS and in a VLF PDS-class. Each time an IMS DPROP component updates control information (for example, the status file), the GMTS is also updated. The updating component invalidates the VLF copy of the GMTS on the local MVS system. Other VLFs in the Sysplex are notified of the change and their GMTSs become invalid too. IMS DPROP components running on other MVS systems in the Sysplex detect that the objects in their VLF space are no longer valid because the VLF copy of the GMTS is invalid.

After updating the DASD copy, IMS DPROP copies the updated GMTS into the local VLF space. However, IMS DPROP does not refresh the VLFs of other MVS systems. When an IMS DPROP component runs on the 'remote' MVS system, it detects that the GMTS has been invalidated but does not know which piece of control information has changed and must delete all the non-PDS class of IMS DPROP objects from the VLF space, reread the GMTS from DASD into VLF and start to populate the VLF space with IMS DPROP control information on a needs-be basis.

Creating and updating the GMTS

Each IMS DPROP system requiring the Sysplex feature, must have a unique member in the GMTS PDS. The member name is the same as the IMS DPROP system name. The member is created in the PDS by the SCU during the processing of an INIT command. Therefore, when the IVP job S*xxx*DPRI is run as part of IMS DPROP system installation, the IMS DPROP-system GMTS member is created.

The GMTS is updated whenever IMS DPROP control information is changed. Update of the GMTS is performed internally by IMS DPROP components.

Refreshing or recreating the VLF PDS

If the IMS DPROP member in the PDS is deleted, you must run the SCU with the INIT command to recreate the GMTS member. The GMTS record is recreated by the next IMS DPROP component attempting to update the GMTS.

The VLF PDS class can be refreshed (for example, after a VLF stop) using the SCU INIT VLF command, or the individual components will refresh the VLF class eventually.

JCL changes for Sysplex IMS DPROP

For Sysplex IMS DPROP systems, each IMS DPROP component or propagating IMS region must include an EKYGMTS DD statement in their JCL. The DD statement is dynamically allocated at system initialization as shown in Figure 16.

```
//EKYGMTS DD DSN=<GMTS PDS data set name>,DISP=SHR
```

Figure 16. EKYGMTS DD statement.

VLF considerations

The GMTS is a PDS class VLF object. For PDS classes the following naming convention applies:

Major name
The concatenation of the volume serial number and the data set name of the GMTS PDS. If the PDS exists but is not on the volume identified at MVS IPL time, (identified implicitly by the system for cataloged data sets or specified explicitly through the VOLUME parameter of the CLASS statement in COFLVF*xx*), VLF does not allow access to that class of objects.

Minor name
The PDS member names form the minor names. Each IMS DPROP system has its own member in the PDS.

One PDS data set can be used by multiple IMS DPROP systems. However, for performance reasons, we recommended that you create a separate PDS for each system to prevent contention among multiple systems for the same data set.

MVG input tables

You can use the MVG input tables to generate propagation requests without DataRefresher. You use programs you have written to access a repository or QMF. Each person defining propagation requests in the MVG input tables must be authorized to update the tables.

The MVG input tables are not considered part of the IMS DPROP directory.

Audit trail table

IMS DPROP records important events in the audit trail which is a System Management Facilities (SMF) data set. To access the information, you run the Audit Extract utility (AUDU). The AUDU utility extracts the audit trail records from the SMF data set and loads them into the audit trail table which is a DB2 table. Query the audit trail table to obtain propagation information.

For more information on the audit trail, refer to the *Reference*.

IMS DPROP operating environment

You might decide to define only one IMS DPROP system. However, if you are doing both production and test work on the same MVS system, you might want to define more than one system.

One IMS DPROP system can support LOG-ASYNC or user asynchronous propagation. If you need to use more than one type of propagation, you must define a separate system, directory, and set of propagation requests for each type of propagation.

One IMS DPROP system can serve only one DB2 system, the DB2 system that contains the IMS DPROP directory tables. If you need to propagate to or from multiple DB2 systems, you must define at least one IMS DPROP system for each DB2 system. Additionally, an IMS dependent or batch region can only propagate with a single IMS DPROP system to or from a single DB2 system.

For asynchronous propagation, you can propagate the same IMS data to more than one DB2 system. Since one IMS DPROP system is connected to only one DB2 system, you need an IMS DPROP system for each propagated DB2 system. You also need to run multiple copies of the Receiver, one for each DB2 system being updated.

You can propagate the same IMS data both LOG-ASYNC and asynchronously. The following sections provide more information about multiple IMS DPROP systems and environments and scenarios for one or multiple IMS DPROP systems, LOG-ASYNC or asynchronous.

Multiple IMS DPROP systems and environments

Each IMS DPROP system has its own directory. The IMS DPROP system is not aware of other IMS DPROP systems and propagation requests defined in the directories of other IMS DPROP systems.

Each system you define has its own set of:
- Directory tables
- VLF objects and class
- Propagation requests
- SQL update modules
- DB2 plans that provide access to both the directory tables and the propagated tables

Each IMS DPROP system also has its own propagated DB2 tables.

When defining JCL and binding DB2 plans for propagating applications or IMS DPROP utilities, be sure to provide consistent definitions. For example, all of the following control information should belong to the same IMS DPROP system:
- IMS DPROP directory tables, accessed through the DB2 plan
- Propagated tables, accessed through the DB2 plan
- Load library, containing the SQL update modules

Multiple IMS DPROP systems can either share the same environment or belong to distinct environments. Each IMS DPROP environment can have its own generation-time parameter values. For example, an environment can have an MVS

SVC number reserved for IMS DPROP use and MVS ROUTCDE used to route IMS DPROP messages to MVS consoles. Each IMS DPROP environment also has its own IMS DPROP load module library.

Usually, all IMS DPROP systems at an installation share the same IMS DPROP environment. However, using distinct IMS DPROP environments and load module libraries allows you to have IMS DPROP systems at different release or maintenance levels.

Scenarios for one or multiple IMS DPROP systems asynchronous

This section describes typical scenarios for using one or more IMS DPROP systems. The scenarios are for the synchronous environment. LOG-ASYNC propagation scenarios are similar but reflect only one-way IMS-to-DB2 propagation and include use of Selectors and Receivers. See “LOG-ASYNC propagation scenarios” on page 82.

If you are implementing both synchronous and asynchronous propagation, you need separate IMS DPROP systems for synchronous and LOG-ASYNC.

Scenario 1

The scenario in Figure 17 is usually used for a test- or production-only MVS system. In this scenario, there is a single IMS environment consisting of one or more IMS subsystems sharing the same RECON data sets, one single IMS DPROP system, and one single DB2 system.

Figure 17. IMS DPROP Scenario 1

If your installation executes both production and test jobs on the same MVS system, we recommend that propagation definitions for production and test jobs be separated into different IMS DPROP directories and systems. See scenarios 2 and 3.

Scenario 2

The scenario in Figure 18 on page 81 shows IMS data that is propagated from one IMS environment, that consists of one or more IMS subsystems sharing the same RECON data sets, to one DB2 system through two or more IMS DPROP systems.

The IMS environment and the DB2 system are used for both test and production data. One IMS DPROP system propagates test data and the other propagates production data.

The propagation requests used to propagate test or production data are defined in the IMS DPROP directory for the IMS DPROP test system. The JCL of the propagating IMS regions used to update test data has a DD statement for the status file of the IMS DPROP test system. The DB2 plan used to execute propagating test programs provides access to both the:

- Directory tables of the IMS DPROP test system
- Propagated test tables

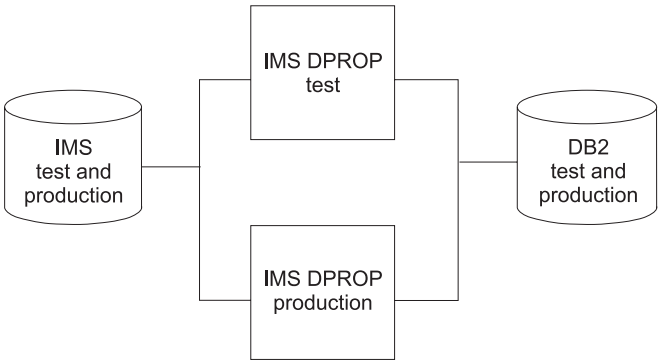


Figure 18. IMS DPROP Scenario 2

Scenario 3

The scenario in Figure 19 shows two completely different operational environments for propagating test and production data.

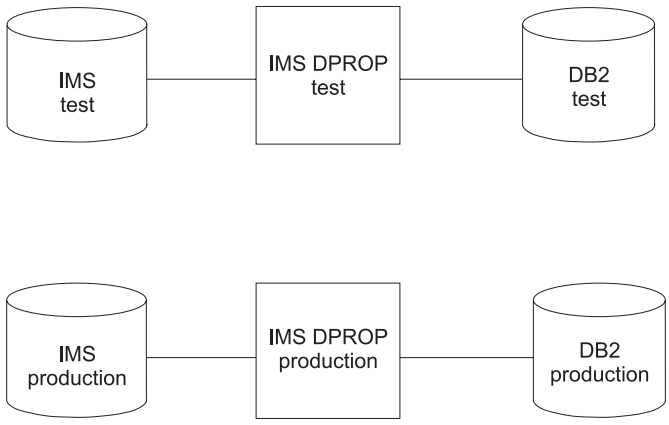


Figure 19. IMS DPROP Scenario 3

Scenario 4

The scenario in Figure 20 on page 82 shows how two IMS environments are connected to one DB2 system through two IMS DPROP systems.

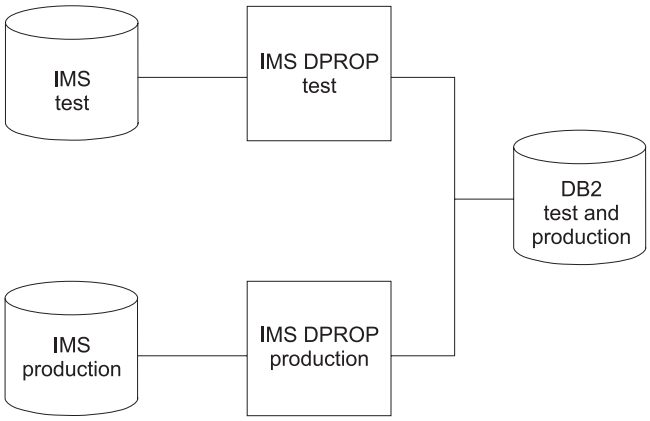


Figure 20. IMS DPROP Scenario 4

Scenario 5

The scenario in Figure 21 shows how one IMS environment, that is used for both test and production jobs, is connected to two DB2 systems through two IMS DPROP systems.

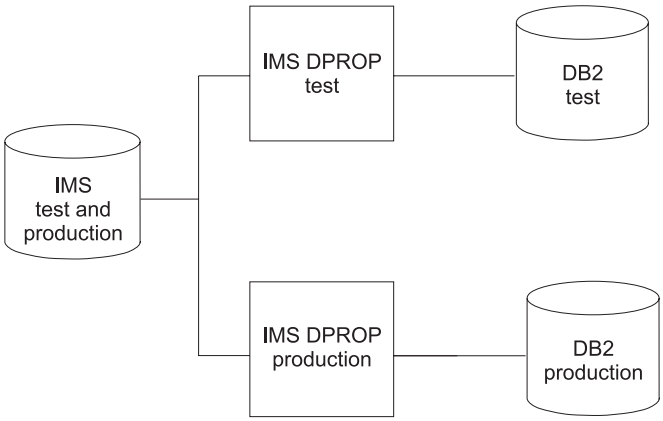


Figure 21. IMS DPROP Scenario 5

LOG-ASYNC propagation scenarios

The section shows scenarios for using one or more IMS DPROP LOG-ASYNC propagation systems. In all the scenarios, IMS with its associated Selectors and DB2 with its associated Receivers can be on the same MVS image, different MVS images with shared DASD, or on remote MVS images.

Scenario 1

The scenario in Figure 22 on page 83 shows the simplest LOG-ASYNC propagation scenario, which can be used for either a test-only or a production-only MVS system. In this scenario, there is a single IMS environment that consists of one or more IMS subsystems sharing the same RECON data sets, one Selector, one Receiver, and one DB2 system.

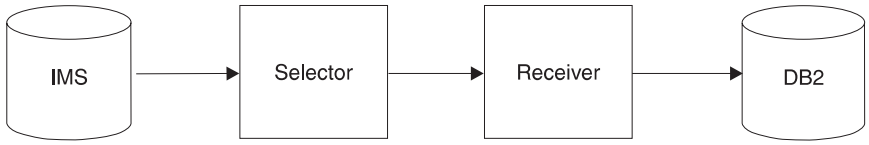


Figure 22. LOG-ASYNC propagation Scenario 1

Scenario 2

The scenario in Figure 23 shows how a single IMS environment that consists of one or more IMS subsystems sharing the same RECON data sets and that is used for both test and production jobs is connected to two DB2 systems using two Selectors and two Receivers.

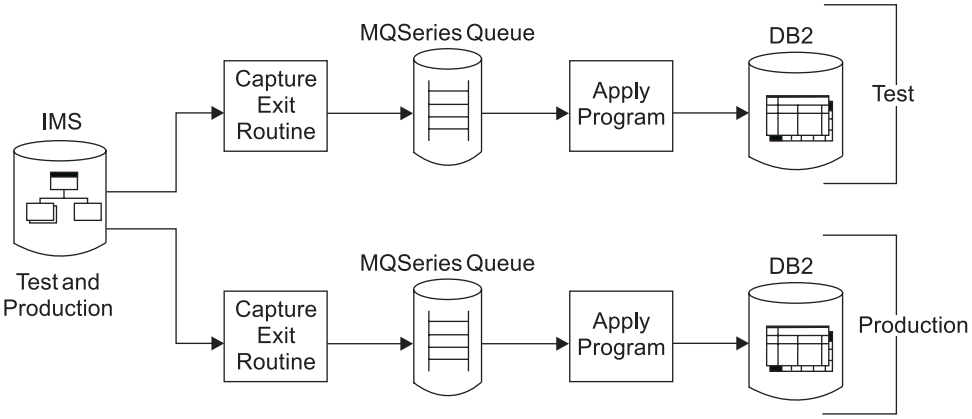


Figure 23. LOG-ASYNC Propagation Scenario 2

Scenario 3

The scenario in Figure 24 shows how two IMS systems, each with its own RECON data sets, is connected to a single DB2 system using two Selectors and two Receivers.

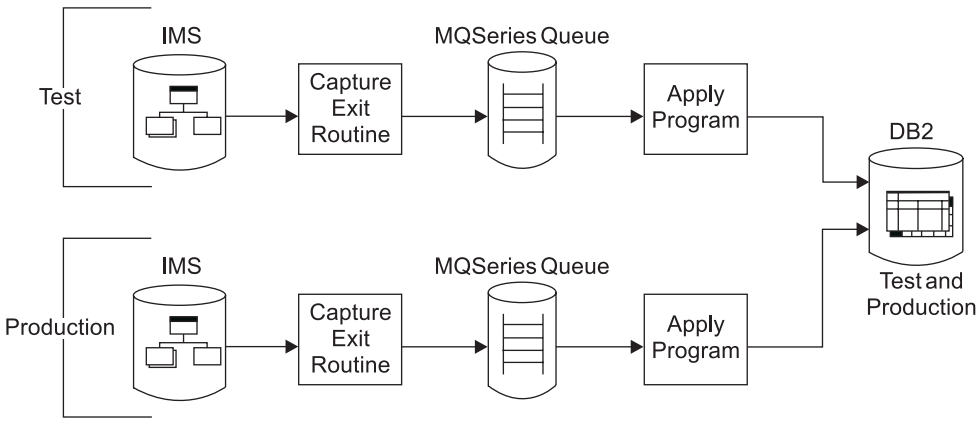


Figure 24. LOG-ASYNC Propagation Scenario 3

IMS environment

This section describes how your IMS environment should be set up before propagation. This section covers:

- Use of DBRC
- Intersystem data sharing
- DBCTL support of changed data capture
- Extended Recovery facility (XRF) considerations
- IMS inserts in load mode
- Database updates with IMS utilities

Use of DBRC

For IMS-to-DB2 propagation, your IMS databases that are to be propagated must be registered in the Database Recovery Control (DBRC).

To maintain the integrity of data, make sure the DBRC share control is in effect. There is no requirement that databases be shared at either the database or block level. DBRC is necessary because:

- For all types of propagation you must ensure that IMS databases are not updated during the IMS extract and DB2 load process because data inconsistency between IMS and DB2 can occur.

  If you are extracting full function IMS databases, you can set the databases to read-only. If you are extracting full function IMS databases with DataRefresher, IMS DPROP checks that the IMS database is read-only during the extract *if* both:
  - The IMS database is registered to DBRC
  - DBRC share control is in effect
- The Selector uses DBRC to locate the logs that contain updates to affected databases. If the databases are not registered to DBRC, they will not be on the RECONs.
- To reduce the possibility of the DBRC deleting SLDS from the RECON before the Selector has a chance to process them, set your recovery period to twice the period between successive runs of the Selector. Use the CHANGE.DBDS DBRC command with a RECOVPD parameter.

All IMS subsystems propagating with the same IMS DPROP system should use the same RECON data sets.

With IMS Version 8, DBRC PRILOG compression has been enhanced. Since DBRC now supports RECON records of unlimited size, compression is attempted whenever an Online Log Data Set (OLDS) is archived. To control this compression activity, use the parameter LOGRET in INIT.RECON. LOGRET is optional and can be used to specify the retention period for log data sets. If you do not explicitly specify this parameter, then the retention period defaults to 1 day (24 hours). You can also use LOGRET option with the CHANGE.RECON control statement to set the retention period. You can view the value for retention in a RECON listing. For the Selector to function correctly, it might be necessary to modify the LOGRET parameter to ensure that the logs remain listed within the RECON (and that compression doesn't occur) long enough for all Selector processing to safely complete, including a safety cushion.

DBCTL support of changed data capture

IMS DPROP supports propagation in a DBCTL environment. For BMP regions, IMS DPROP supports all types of propagation. For CICS transactions, IMS DPROP supports:

- LOG-ASYNC propagation.
- User asynchronous propagation, implemented with the IMS Asynchronous Data Capture function only.

  Because IMS DPROP is not called by IMS Data Capture and DB2 Data Capture when updates are made through CICS transactions executing in a DBCTL system, the updates cannot be propagated synchronously. However, IMS databases and DB2 tables that are propagated synchronously can be accessed in read-only mode by CICS applications.

Extended Recovery Facility (XRF) considerations

For LOG-ASYNC propagation, the XRF takeover should be transparent.

The Selector retrieves the IMS updates from the IMS logs, using the DBRC RECON data sets to locate the logs.

An XRF takeover does not affect the Selector's ability to locate and process the correct logs, because the primary IMS subsystem and the backup XRF subsystem:

- Use the same RECON data sets.
- Use the same log data sets.
- Write the same IMS subsystem ID (SSID) to the RECON data sets. You must define the SSID in the SCF as either a default SSID or as the SSID for the propagation group.

You do not need to change the JCL for running the Selector after an XRF takeover because it does not matter whether the Selector invokes DBRC from the IMS RESLIB for the primary IMS subsystem or the backup XRF subsystem.

IMS inserts in load mode

When doing IMS-to-DB2 propagation for:

LOG-ASYNC propagation

IMS DPROP might or might not propagate inserts done with PCBs having PROCOPT=L or LS specified. The DBRC:

- Does not allocate the log in the PRILOG record of the RECONs if all the updates in a log relate to inserts done in load mode. Therefore, the Selector cannot locate the log.
- Allocates the log in the PRILOG record of the RECONs if some of the updates in a log relate to inserts done in a mode other than load mode. Therefore, the Selector can locate the log and writes all committed 9904 (update) records, including those done in load mode to the PRDS.

If you do not want the inserts propagated, either:

- Do not alter the IMS DBDs to activate the EXIT keyword until the load operation is complete
- If the EXIT keyword has already been activated:
  1. Quiesce the IMS databases after the load has completed
  2. Synchronize the relevant DB2 tables with the IMS databases
  3. Create quiesce timestamps for the IMS databases
  4. Assign the quiesce timestamps as start timestamps for the IMS databases

User asynchronous propagation

If you use the IMS Asynchronous Data Capture function to log updates, DBRC:

- Does not allocate the log in the PRILOG record of the RECONs if all the updates in a log relate to inserts done in load mode
- Allocates the log in the PRILOG record of the RECONs if some of the updates in a log relate to inserts done in a mode other than load mode

Database updates with IMS utilities

When segments are inserted into a database using any IMS database utilities, IMS DPROP is not called. IMS DPROP does not propagate IMS segments during database reload with the IMS HD Reorganization Reload utility. Other database utilities, including IMS HISAM Reload, Database Recovery, and DEDB Direct Reorganization utilities, also are not called.

DB2 environment

This section describes how your DB2 environment should be set up before propagation. Basically:

- An IMS dependent or batch region can do synchronous propagation to or from only one DB2 system
- A segment can be propagated to or from only one DB2 system

CICS environment

In a CICS environment with DBCTL, IMS DPROP supports:

- LOG-ASYNC propagation
- User asynchronous propagation with IMS LOG-ASYNC Data Capture function only

CICS environments with local DL/I are not supported.

Reducing operational risks using ERROPT=IGNORE

You can reduce some of the risks of propagation failures affecting the availability of your applications by defining propagation requests with ERROPT=IGNORE. This option is usually used in only test systems.

Part 3. Setting Up for Data Propagation

Chapter 5. Setting up your systems for LOG-ASYNC propagation

This chapter describes how to set up IMS and DB2 for LOG-ASYNC and user asynchronous propagation. Activities described in this chapter are:

- Creating or changing DBDs
- Creating DB2 tables, if needed
- Binding plans
- Creating, modifying and deleting propagation groups
- Adding SSIDs to propagation groups
- Establishing the start conditions for propagation

Creating or changing DBDs

DBDs are used during the process of creating propagation requests. If the propagated IMS databases do not yet exist, you must create their IMS DBDs. If the DBDs already exist, you might need to makes changes to them. For directions on how to change a DBD, refer to the *IMS/ESA Utilities Reference: System*. This section describes what you must do to accommodate IMS DPROP.

To create IMS DBDs if IMS databases are propagated in an IMS *online* environment, you must:

1. Define the DBDs to the online IMS system.
2. Run IMS ACBGEN.

When you make changes to your DBDs for IMS DPROP, run DBDGEN. Reasons for changing your existing DBDs through a DBDGEN are:

IMS-to-DB2 propagation

You must specify an EXIT= keyword to identify the propagated segments to IMS. The EXIT keyword is specified in the DBD macro or in the SEGM macros of the DBD defining the physical database. Specifying the EXIT keyword activates the IMS Data Capture function. You must specify the EXIT= keyword before propagation requests are created. See “EXIT Keyword (IMS-to-DB2)” on page 90 for examples of how to specify the EXIT keyword.

You also must verify that any existing IMS delete rules comply with the IMS DPROP restrictions described in “IMS logical relationship rules” on page 39. You might need to modify your delete rules, and make changes to the logic of your application programs.

All types of propagation

You can specify a VERSION= keyword in the DBD macro. Depending on IMS DPROP generation options specified during installation, either the RUP or HUP (for synchronous) might validate the version to reduce errors resulting from inconsistent DBD libraries and IMS DPROP directories.

After a DBD is changed, you must run ACBGEN if the database is referred to in an online IMS environment. Refer to IMS/ESA Utilities Reference: System for more information about the ACBGEN and DBDGEN processes.

The following sections describe how you use:

- EXIT keyword
- VERSION keyword

EXIT Keyword (IMS-to-DB2)

For IMS-to-DB2 propagation, the DBD defining the physical database must include an EXIT keyword.

On the EXIT keyword you specify whether or not:

- The Asynchronous Changed Data Capture function writes captured data to the IMS log.
- IMS calls one or more IMS Data Capture exit routines with the captured data.

On the EXIT keyword:

LOG-ASYNC propagation
Specify LOG. You can specify LOG either with or without the name of an IMS Data Capture exit routine.

User asynchronous propagation
Specify your own IMS Data Capture exit routine's name. Your exit routine is often referred to as the *sender program*.

User asynchronous propagation with the IMS Asynchronous Data Capture function
Specify LOG. You can specify LOG either with or without the name of an IMS Data Capture exit routine.

You also use the EXIT keyword to specify data options that determine the type of data to be captured. The data options can be different for each exit routine. See "Specifying data options on the EXIT keyword" on page 91.

You can specify multiple exit names, so you can propagate the same segment both synchronously (using EKYRUP00) and user asynchronously (using your sender program). Specifying multiple names also lets you propagate the same segment to both DB2 using EKYRUP00 and other IMS databases using an exit you write. If multiple exits are defined in the EXIT keyword, they are called in the order in which they are defined.

You can also specify LOG with EKYRUP00 as an exit name, so you can propagate the same segment both synchronously with EKYRUP00 and user asynchronously with the IMS Asynchronous Data Capture function.

The following sections discuss:

- Specifying the EXIT keyword at either DBD or SEGM level
- Specifying the EXIT Keyword with logical child segments
- Specifying data options on the EXIT keyword
- Examples of the EXIT keyword

Specifying the EXIT keyword

Specify the EXIT keyword at either the DBD or SEGM level.

Specifying EXIT in the DBD macro: If all or most segments in a database are to be propagated, it is convenient to specify the EXIT keyword in the DBD macro. With an EXIT parameter, the DBD macro calls the IMS Data Capture function when changes are made to any segment types in the database. Propagation is then done for those segment types that have propagation requests defined in the IMS DPROP directory. Segments having no propagation requests defined are not propagated. You can specify EXIT=NONE in the SEGM macro to override an EXIT keyword specified in the DBD macro.

Specifying EXIT in the SEGM macro: If only a few segments in a database are to be propagated, specify the EXIT parameter in the SEGM macros defining the segments to be propagated. You can improve performance by limiting calls to the IMS Data Capture function. Specifying an EXIT parameter in a SEGM macro overrides an EXIT keyword specified in a DBD macro.

If you are using the generalized mapping cases and your propagation request specifies PATH=DENORM or ID, then you must also specify an EXIT parameter for those physical parent/ancestor segments that contribute PATH data.

If a DBD or SEGM macro specifies an EXIT keyword and no propagation requests are defined in the IMS DPROP directory, RUP returns without doing any propagation and without indicating any errors. Therefore, you can make the necessary changes to the DBD before defining propagation requests.

Specifying the EXIT keyword with logical child segments

For physically- or virtually-paired logical child segments, only one of the two logical child segment types should be propagated.

- For virtual pairing, the physical logical child of the pair must be propagated, not the virtual
- For physical pairing:
  - If either of the two children in the pair has propagated dependent segments, that child should be propagated
  - If neither of the two children in the pair has propagated dependent segments, it doesn't matter which segment of the pair is propagated

Propagate only one of the two segment types of the pair by providing an EXIT keyword for only one segment type of the pair, or by defining propagation requests for only one segment of the pair.

Specifying data options on the EXIT keyword

The EXIT keyword supports a set of data options that determine what data is passed to the Data Capture exit routine and logged. Each exit routine specified on the EXIT keyword has its own set of data options.

During propagation request definition, IMS DPROP validates that data options are compatible with propagation request definitions. For user asynchronous, IMS DPROP validates the data options for whatever is specified last on the EXIT keyword. Therefore, if you are providing multiple specifications in an EXIT keyword, for example, specifications for multiple exit routines, it is recommended that specifications for user asynchronous propagation come last.

For generalized mapping cases, you can generally omit data options whose defaults are KEY, DATA, NOPATH (CASCADE, KEY, DATA, NOPATH). For propagation request definitions that include PATH=ID or DENORM or for some mapping case 2 propagation requests, you must override the default NOPATH option by specifying the PATH data option.

For user mapping cases, you might have different requirements for which you need to use different data options.

IMS DPROP supports PATH/NOPATH and CASCADE/NOCASCADE options. IMS DPROP does not support IMS NOKEY and NODATA options.

PATH/NOPATH: The PATH/NOPATH data option specifies whether or not data from each segment in the hierarchic path from the physical root is to be passed to the Data Capture exit routine.

NOPATH does not pass data to the Data Capture exit routine. NOPATH is the default but is not always valid with IMS DPROP. PATH passes data to the Data Capture exit routine. PATH is always valid for IMS DPROP.

For generalized mapping cases, if your propagation requests specify PATH=DENORM or ID, be aware of the following DBD requirements:

- For the entity segment and any extension segment, the EXIT keyword must be specified with KEY, DATA, and PATH data options in effect.
- For each physical parent and physical ancestor of the entity segment that contributes path data, the EXIT keyword must be specified with KEY and DATA options in effect. With the exception of the highest-level physical parent/ancestor contributing path data, the PATH data option must also be in effect.

For generalized mapping case 2 propagation requests, if non-key fields of the entity segment are mapped to the DB2 primary key or included in the WHERE clause, KEY, DATA, and PATH data options must also be specified.

For other generalized mapping cases, a PATH specification is not useful and increases the path length of your propagating application.

For user mapping cases, depending on the mapping logic you are using, you might have to specify PATH. For example, you should use PATH if a Propagation exit routine propagates data from both the changed segment and its physical ancestors.

CASCADE/NOCASCADE: The CASCADE/NOCASCADE data option specifies whether RUP is called during cascading IMS deletes. Cascading IMS deletes occur when the physical parent or a physical ancestor segment is deleted.

When NOCASCADE is specified for a particular segment type, RUP is *not* called during a cascading delete of that segment type. The RUP is not called for a segment type whose physical parent or a physical ancestor is deleted.

When CASCADE is in effect for a particular segment type, RUP is called during a cascading delete of that segment type. The logical parent or logical child is deleted when cascading deletes cross an IMS logical relationship causing the RUP to be called regardless of the CASCADE/NOCASCADE option.

CASCADE is always valid for IMS DPROP, and is the IMS default. CASCADE includes suboptions whose default values are (CASCADE, KEY, DATA, NOPATH). For generalized mapping cases, you usually omit the suboptions. However, you must explicitly specify the PATH data suboption for propagation request definitions of PATH=ID or DENORM and for some mapping case 2 propagation requests.

When valid, the NOCASCADE option can sometimes reduce path length for propagation of deleted segments. For purposes of propagation, NOCASCADE is only valid when either:

- The segment being propagated is an extension segment under generalized mapping case 2
- The IMS parent/child relationship is matched by a DB2 parent/child RIR in which ON DELETE CASCADE is specified

When Using LOG with CASCADE

If IMS is unable to log all of a 9904 record, due to an excessively large amount of data in the record, it sets the error flag in the 9904 record. If the Selector reads a 9904 record whose error flag is set, it issues a warning message and continues processing. The error situation is unusual, but can occur with Cascade deletes if PATH data is specified, particularly with CICS/DBCTL.

To avoid the error situation where a record containing an excessively large amount of data is written to the log, you should ensure that only the concatenated key and the segment data, not the path data, is written to the log by specifying on the DBD:

- NOCASCADE with PATH or NOPATH if one or more of your propagation requests specify path data and you have defined RIRs on the target DB2 tables

  ```
  EXIT=(*,KEY,DATA,PATH(NOCASCADE),LOG)
  ```

  Or

  ```
  EXIT=(*,KEY,DATA,NOPATH(NOCASCADE),LOG)
  ```

- NOPATH and CASCADE, if none of your propagation requests specify path data

  ```
  EXIT=(*,KEY,DATA,NOPATH(CASCADE,KEY,NODATA,NOPATH),LOG)
  ```

If you want to propagate path data without defining RIRs for the target DB2 tables, you can specify:

```
EXIT=(*,KEY,DATA,PATH(CASCADE,KEY,NODATA,PATH),LOG)
```

so that your propagation requests can propagate path data. However, specifying path data and cascade deletes can result in large volumes of log data for a single record (for example if the root segment is deleted). Data must be stored in memory before being written to the log and, in extreme circumstances, it can exceed the size of the available memory. IMS can only log the portion of the record that is in memory and sets the error flag for the record. When the Selector is executed and detects the error flag, it issues a warning message and continues processing.

If you receive a warning, you must do one of the following:

- Specify NOCASCADE on the EXIT keyword and define RIRs for the target DB2 tables
- If it is not essential that you specify path data, redefine your propagation requests so that none of them specify path data and specify NOPATH and CASCADE on the EXIT keyword

PATH|NOPATH suboption of CASCADE: CASCADE includes a PATH|NOPATH suboption. The PATH|NOPATH data option specifies whether or not data from each segment in the hierarchic path from the physical root is to be passed to the Data Capture exit routine.

IMS DPROP has the same requirements for the PATH|NOPATH suboption as for the PATH|NOPATH options described in “PATH/NOPATH” on page 92. The default is NOPATH.

IMS DPROP requires a PATH option or suboption if your propagation request specifies PATH=DENORM or ID. A PATH option or suboption might also be required for some mapping case 2 propagation requests as described in “PATH/NOPATH” on page 92.

NODATA suboption of CASCADE: IMS DPROP supports the combination of (CASCADE, KEY, NODATA) only for user mapping and for generalized mapping cases 1 and 2 in cases where both the:

- DB2 primary key of the propagated table is mapped only from the IMS fully concatenated key of the changed segment
- WHERE clause includes only fields from the IMS fully concatenated key

Examples of the EXIT keyword

For more information on the DBD macro's EXIT keyword, refer to IMS/ESA Utilities Reference: System and the IMS DPROP *Reference*.

In this section, examples:

1 and 2
Show how to specify both LOG-ASYNC and user asynchronous propagation of the same segment type.

3 to 6 Do not have data options explicitly specified, but default to KEY, DATA, NOPATH, (CASCADE, KEY, DATA, NOPATH). These defaults are acceptable if no propagation request is defined with PATH=ID or DENORM.

7 to 10
Explicitly specify PATH both as a data option and as a data suboption of CASCADE. Examples 7 to 10 support all propagation request definitions, including propagation requests defined as PATH=ID or DENORM.

The following examples apply to LOG-ASYNC or user asynchronous propagation.

Example 1 (any combination of synchronous, LOG-ASYNC and user asynchronous propagation)

```
SEGM NAME=segname,PARENT=psegname,BYTES=nn,EXIT=(EKYRUP00,LOG,NOCASCADE)
```

Shows how to propagate the segment type:
- Synchronously with EKYRUP00
- User asynchronously with the IMS Asynchronous Data Capture function
- LOG-ASYNC with no cascade deletes

Example 2 (any combination of synchronous and user asynchronous propagation)

```
SEGM NAME=segname,PARENT=psegname,BYTES=nn,EXIT=(EKYRUP00,NOCASCADE),
(sender,NOCASCADE)
```

Shows two different IMS Data Capture exit routines:
- EKYRUP00 propagates the segment synchronously.
- The sender program is specified as the last exit routine to do user asynchronous propagation. As recommended, specifications for user asynchronous propagation are defined last (the sender program follows EKYRUP00).

Example 3 (any combination of LOG-ASYNC and user asynchronous propagation using ACDC)

```
DBD NAME=dbname,ACCESS=...,EXIT=(*,LOG)
```

Example 4 (user asynchronous propagation using CDC exit)

```
DBD NAME=dbname,ACCESS=...,EXIT=(sender)
```

Example 5 (any combination of synchronous, LOG-ASYNC, and user asynchronous propagation using ACDC)

```
DBD NAME=dbname,ACCESS=...,EXIT=(EKYRUP00,LOG)
```

Example 6 (user asynchronous only using CDC exit)

```
DBD NAME=dbname,ACCESS=...,EXIT=((EKYRUP00),(sender))
```

Example 7 (any combination of LOG-ASYNC and user asynchronous propagation using ACDC with path data)

```
DBD NAME=dbname,ACCESS=...,EXIT=(*,LOG,PATH,(CASCADE,PATH))
```

Specifies EXIT=(*,LOG). The asterisk (*) specifies to IMS that no IMS Data Capture exit routine is to be called. LOG specifies that the IMS Asynchronous Data Capture function is to write captured data to the IMS log.

Example 8 (user asynchronous only using CDC exit with path data)

```
DBD NAME=dbname,ACCESS=...,EXIT=(sender,PATH,(CASCADE,PATH))
```

Specifies the sender program as the exit routine to do user asynchronous propagation.

Example 9 (any combination synchronous, LOG-ASYNC, and user asynchronous propagation using ACDC with path data)

```
DBD NAME=dbname,ACCESS=...,EXIT=(EKYRUP00,LOG,PATH,(CASCADE,PATH))
```

Specifies synchronous propagation and either LOG-ASYNC propagation, user asynchronous propagation, or both, of the same database:

- The EKYRUP00 exit routine does synchronous data propagation
- LOG specifies that changed data is to be written by the IMS Asynchronous Data Capture function to the IMS log for LOG-ASYNC and user asynchronous propagation.

Example 10 (any combination synchronous and user asynchronous propagation using ACDC with path data)

```
DBD NAME=dbname,ACCESS=...,EXIT=((EKYRUP00,PATH,(CASCADE,PATH)),
                                 (sender,PATH,(CASCADE,PATH)))
```

Specifies synchronous propagation and user asynchronous propagation of the same database. Two different IMS Data Capture exit routines are specified. EKYRUP00 propagates the database synchronously. The sender program is the last exit routine to do user asynchronous propagation. As recommended, specifications for user asynchronous propagation are defined last (the sender program follows EKYRUP00).

After changing the DBDs, you must perform a DBDGEN. The IMS DBDLIB created is used as input to the IMS DPROP propagation request creation process.

The DLIBATCH IMS job begins to log data for ACDC databases as soon as the DBDGEN has been performed. A DBBBATCH IMS job also requires an ACBGEN. An online IMS system requires an ACBGEN in combination with a system quiesce and restart, or a database quiesce and online change, in order for the ACDC function to become active.

Specifying the VERSION keyword

You can specify a VERSION keyword in the DBD macro to associate a version ID of your choice with the DBD. If you do not specify a VERSION keyword, IMS generates a version ID based on a timestamp.

Depending on the DBDV keyword specified during IMS DPROP generation, the RUP validates the version ID during propagation to reduce errors resulting from inconsistencies between DBD libraries and IMS DPROP directories.

The DBDV keyword of the IMS DPROP generation macro EKYGSYS specifies the offset and length of the part of the version ID that is to be validated. If the length is zero, no validation is performed. If the length is not zero, the MVG stores the version ID from the DBD in the IMS DPROP directory during propagation request definition. RUP verifies the version ID received from the IMS Data Capture function against the version stored in the IMS DPROP directory.

If you use the default IMS DPROP generation option, the RUP validates the full version ID. However, during IMS DPROP generation, IMS DPROP allows you to identify the portion of the version ID to use for validation. For example, you might want to use part of the ID to distinguish between DBD changes affecting propagation and those that do not, such as randomizing parameters.

The following examples illustrate how you can use the VERSION keyword.

- When an IMS DPROP-related DBD change is made, you can alter the part of the version ID that was defined to IMS DPROP for version checking. After performing the DBDGEN, you must:
  1. Modify the propagation request definitions that are affected by the change
  2. Regenerate all propagation requests for the altered DBD
  3. If the DBD change has an effect on mapping or propagation, re-extract the affected data from the:
     - IMS database (IMS-to-DB2 propagation only)
     - DB2 tables (DB2-to-IMS propagation only)
- If you make a change to the DBD that does not affect propagation or mapping, the IMS DPROP-specific part of the version ID should not change.
- If the DBD change includes a change to the IMS DPROP-related part of the version ID but has no impact on mapping or propagation, propagation requests must be regenerated; however, you do not need to re-extract the data. You can use the PERFORM=BUILDONLY propagation parameter when you regenerate the propagation request with DataExtractor.
- If you are using VLF, you must enter the SCU control statement INIT VLF as described in the *Reference*.

For more information on the VERSION keyword of the DBD macro, refer to *IMS/ESA Utilities Reference: System* and the IMS DPROP *Reference*.

Creating DB2 tables

DB2 target tables (model target tables) must exist before you create the IMS DPROP propagation requests. For local and remote MVS images, target tables must be created on the target MVS image.

To create your propagated DB2 tables and establish RIRs among them you need to code and execute the appropriate SQL statements. For more information on the SQL CREATE TABLE statement, refer to DB2 Administration Guide.

For IMS DPROP, you must:

- Define columns
- Determine if you are to use qualified or unqualified tables

Specifying columns

Define primary key columns as NOT NULL or NOT NULL WITH DEFAULT. Columns that are not mapped from IMS, or that are mapped from IMS fields that may validly

be absent at propagation time, should be defined to permit null values or as NOT NULL WITH DEFAULT. You can use null or NOT NULL WITH DEFAULT for:

- Fields of IMS extension segments using mapping case 2
- Fields of variable-length segments
- Data mapped with a user mapping case

Table qualification

If you are creating propagation requests with *qualified* table names, define the tables before you create propagation requests. If you are creating propagation requests with *unqualified* table names, then you must create *model* tables before you create propagation request.

Binding the Receiver and other plans

For LOG-ASYNC propagation, you need to bind Receiver plans and, if applicable, the DB2 plans of your propagating application programs. The plans must at least provide access to the IMS DPROP directory tables for RUP.

The bind of the application program plan can be done with or without use of the package bind function.

Binding plans with DB2 package bind

Using the DB2 package bind function makes administration of your DB2 plans easier. If you use package bind, you do not to rebind the DB2 plans of your propagating applications after creation of or changes to propagation requests and your exit routines.

To use package bind, you must bind the following DBRMs as DB2 packages:

- DBRMs of IMS DPROP modules accessing the IMS DPROP directory. This package bind is done as part of the generation and installation of IMS DPROP.
- DBRMs of the IMS DPROP SQL update modules. One SQL update module exists for each propagation request in a generalized mapping case. Each SQL update module has a DBRM. The package bind of the DBRMs is done later when you create the propagation requests; the package bind can be automatically done by MVG as part of propagation request creation.
- DBRMs of your IMS DPROP exit routines, for example Propagation exit routines, that issue SQL update statements. Usually you bind the package of the DBRMs after precompiling and compiling of your exit routines.

When binding the plan of your propagating applications, you list in the PKLIST keyword of the BIND command the names of the package collections containing the preceding DB2 packages.

Details about binding propagating applications are discussed in Chapter 8, “Binding and administering plans,” on page 127.

Binding plans without DB2 package bind

To bind the plans of your propagating applications, you list in the MEMBER keyword of the BIND command the names of the RUP and HUP DBRMs that access the IMS DPROP directory.

Later, if you make changes you must re-bind the plans of your propagating applications. After creating or changing your propagation requests and exit routines that issue SQL statements, you must include in the plans the DBRMs of:

- IMS DPROP SQL update modules
- Your IMS DPROP exit routines, for example Propagation exit routines, that issue SQL update statements

If no updates are made until after propagation requests are defined, then the bind is optional.

Details about binding propagating applications are discussed in Chapter 8, “Binding and administering plans,” on page 127.

Creating, modifying, and deleting propagation groups

After creating propagation requests as described in Chapter 6, “Defining and changing propagation requests,” on page 101, collect the propagation requests into propagation groups before running the Selector. As explained in detail in “Receivers” on page 149, you do not directly define a set of propagation requests to a propagation group. Instead you assign propagation requests to Receivers. Use the SCU to create Receivers.

Use the following SCU control statements:

CREATEREC
: Creates a Receiver to process a propagation group and stores the details in the IMS DPROP directory’s Receiver control table (RCT)

ASSIGNPR
: Assigns one or more propagation requests to a Receiver and stores the details in the IMS DPROP directory’s PR control table (PRCT)

Related Reading: Refer to the *IMS DataPropagator for z/OS Reference* for details of the SCU CREATEREC and ASSIGNPR control statements and the associated DELETEREC and DELETEPR control statements.

Before you run the Selector, ensure that definitions of all propagation groups to be processed on the current execution are stored in the SCF. The Selector uses these definitions to determine which IMS source updates should be selected from the IMS logs.

You can update the Selector control file either automatically or manually. To manually update the SCF, create the required SCF control statements and use the SCF Apply utility to apply them to the SCF. To automatically update the SCF, use the Group Unload utility (Group Unload Utility) and the SCF Compare and Apply utilities:

- At the Receiver site, use the GUU SELECT GROUP control statement to specify which propagation groups are to be stored in the Selector control file. The GUU extracts details of the IMS sources for the specified propagation groups from the IMS DPROP directory tables and writes them to the Group Definitions File.

  If the Selector and the Receiver are on different MVS images, without shared DASD, you must transport the Group Definitions File from the Receiver MVS image to the Selector MVS image.
- At the Selector site, you use:
  – The SCF Compare utility to compare the contents of the group definitions file and the Selector control file and to create SCF control statements that create or modify propagation groups in the Selector control file. You can modify the control statements before submitting them, if needed.

– The SCF Apply utility to run the SCF control statements that match the propagation group definitions in the SCF with those in the IMS DPROP directory.

By following this automated method, your input is validated as much as it can be and administration time is reduced. Figure 25 shows the inputs to and outputs from the GUU, SCF Compare utility, and SCF Apply utility.

Related Reading: Refer to the *IMS DataPropagator for z/OS Reference* for details of the GUU and the SCF Compare and Apply utilities.

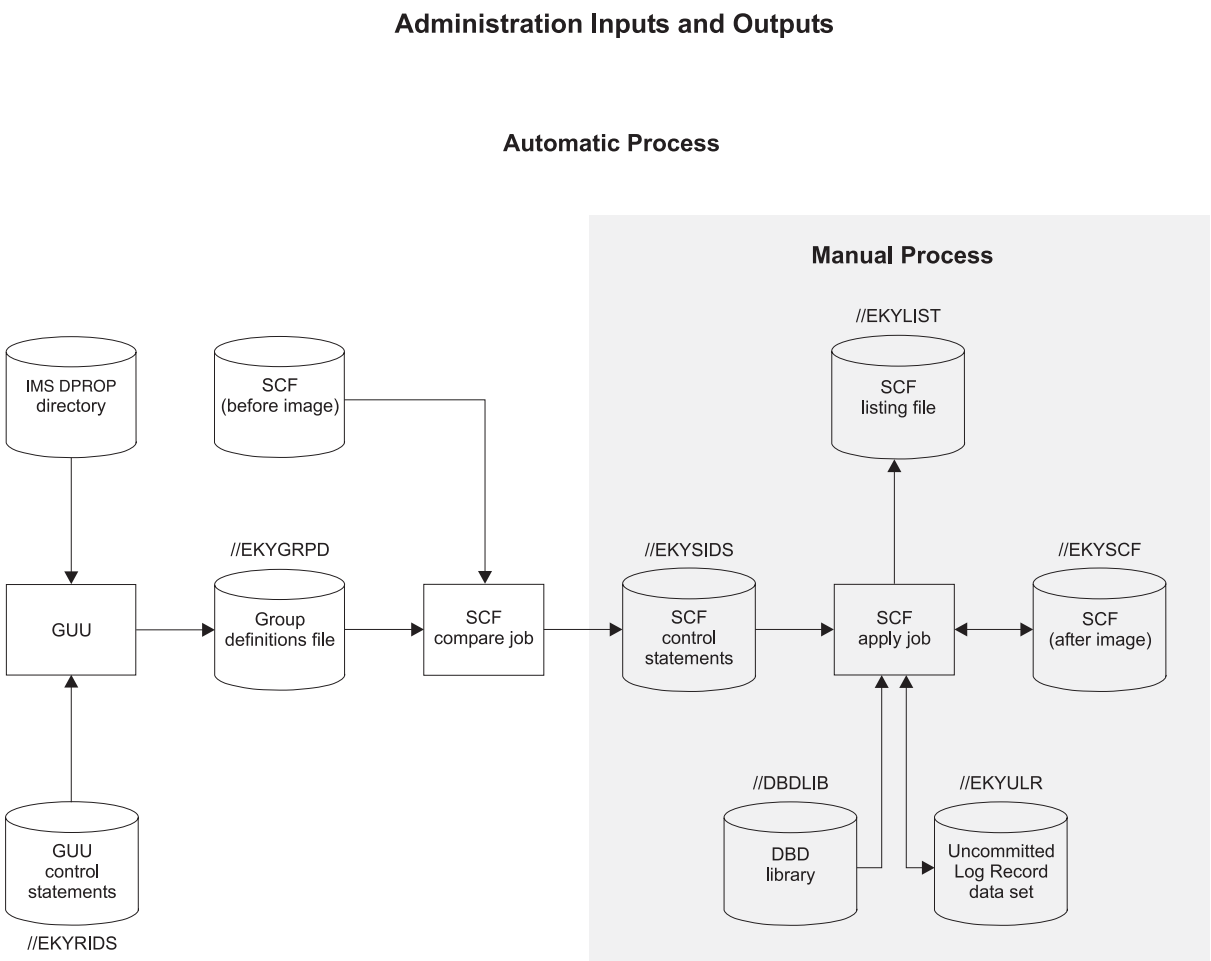


Figure 25. SCF Administration Inputs and Outputs

Adding SSIDs to propagation groups

Use the SCF Apply utility and the ADDSSID control statements to manually define:

- A default set of IMS subsystem IDs (SSIDs) for the IMS DPROP environment, if required
- A set of IMS SSIDs for each propagation group, if required

Related Reading: Refer to the *IMS DataPropagator for z/OS Reference* for details of the SCF Apply utility and the ADDSSID control statements.

Establishing the start conditions for LOG-ASYNC propagation

Before you can start LOG-ASYNC propagation, you must:

1. Use the SCU READON control statement to quiesce all IMS databases that are involved in LOG-ASYNC propagation. Or you must take DEDBs offline.
2. At any time when the IMS databases are quiesced, use the SCU CREATETSM QUIESCE control statement to create one or more SCF 0202 (database/quiesce timestamp) records for each database.
3. For each database, use the SCU ASSIGNTSM control statement to assign one of the 0202 (database/quiesce timestamp) records as a propagation group/database propagation start timestamp in an SCF 0302 (propagation group/database) record. The NEWSTART flag of the 0302 record is set to `Y`.
4. Optional: Use the SCU CREATETSM STOP control statement to create a 0305 (propagation group/stop time) record for each propagation group. If you do not specify a stop time for a propagation group, the Selector can determine a stop time for you, as described in "Considerations for Selector-determined stop time (STOP=INTERIM)" on page 159.
5. Carry out a full extract of the IMS databases and load the data into the DB2 tables, as described in Chapter 9, "Extracting and loading data for IMS-to-DB2 propagation," on page 137.

   To establish a recovery point:

   a. Do a full extract of the IMS databases
   b. Take an image copy of the IMS databases
   c. Load the data into the DB2 tables
   d. Take an image copy of the DB2 databases
   e. Take a copy of the SCF and the ULR data set
6. Use the SCU READOFF control statements to restart all IMS databases that are involved in LOG-ASYNC propagation. Or you can bring DEDBs back on line.

Related Reading: Refer to the *IMS DataPropagator for z/OS Reference* for details of the SCU control statements.

Chapter 6. Defining and changing propagation requests

Creating a propagation request involves specifying the IMS fields (or data elements), IMS segments, DB2 columns, and DB2 tables to be propagated. The information specified associates keys and data from IMS databases to DB2 tables. You can code propagation requests either by using both DataRefresher and IMS DPROP and defining propagation requests with DataRefresher, or by defining propagation requests in the MVG input tables.

To define a propagation request in the MVG input tables, you can provide an application that extracts the necessary information from a dictionary system, or you can use QMF or SPUFI. If you use DataRefresher to build your propagation requests, you do not need to create MVG input tables.

This chapter covers:
- Defining propagation requests using DataRefresher
- Defining propagation requests using the MVG Input Tables
- Propagation parameters
- Deleting a propagation request
- Replacing a propagation request
- Rebuilding a propagation request
- Revalidating propagation requests

Defining propagation requests using DataRefresher

Use both IMS DPROP and DataRefresher to:
- Define IMS DPROP propagation requests using DataRefresher User Input Manager (UIM) commands
- Do the IMS extract and DB2 load process with the DataRefresher Data Extract Manager (DEM) and the DB2 LOAD utility

See the DataRefresher library for details of the DataRefresher UIM and DEM.

This section describes the definition of propagation requests with DataRefresher UIM; see Chapter 9, “Extracting and loading data for IMS-to-DB2 propagation,” on page 137, for a description of the extract and load with DataRefresher DEM.

By combining IMS DPROP and DataRefresher, you can use the same mapping definitions for data propagation and extract and load. The mapping and data conversions done by IMS DPROP for generalized mapping cases during propagation are a compatible subset of the mapping and conversions done by DataRefresher during the extract and load. Compatibility is important to avoid propagation failure and data inconsistency.

You can also use DataRefresher to provide mapping definitions for one-way DB2-to-IMS synchronous propagation, or even if you do not intend to extract IMS data with DataRefresher.

When DataRefresher has an extract request for which data propagation is to be done, the IMS DPROP Map Capture exit (MCE) validates the extract request, which is then called a propagation request.

Before defining propagation requests with DataRefresher, you need to describe the IMS data to be extracted and propagated to DataRefresher. Use the following DataRefresher UIM commands:

- CREATE DATATYPE commands, if you need to use Field exit routines. See "CREATE DATATYPE command" on page 102.
- CREATE DXTPSB commands with DXTPCB, SEGMENT, and FIELD statements. These statements describe your IMS databases, segments, and fields to DataRefresher. See "CREATE DXTPSB command" on page 102.
- CREATE DXTVIEW commands, which identify a hierarchical path of the IMS database used as input to the DataRefresher extract process. See "CREATE DXTVIEW command" on page 103.

Define the propagation requests by providing a SUBMIT DataRefresher UIM command for each propagation request to be defined, with a corresponding EXTRACT statement. See "SUBMIT command and EXTRACT statement" on page 104. Using the UIM command and an EXTRACT statement identifies the propagated DB2 table and describes which IMS segments or fields are to be mapped to which DB2 column. During processing of the SUBMIT command, DataRefresher calls the IMS DPROP Map Capture exit (MCE), which validates the propagation request.

Figure 26 on page 106 illustrates the process. For complete information on using DataRefresher commands to define propagation requests, refer to the *IMS DataPropagator for z/OS Reference*. This section provides more information on:

- The CREATE DATATYPE command
- The CREATE DXTPSB Command
- The CREATE DXTVIEW command
- The SUBMIT command and EXTRACT statement
- DataRefresher and user mapping cases

CREATE DATATYPE command

CREATE DATATYPE commands are optional. Use them only if you intend to use Field exit routines because:

- Your IMS database contains fields in formats not supported directly by IMS DPROP and DataRefresher
- You want to perform data conversions that are not directly supported by IMS DPROP and DataRefresher

Each CREATE DATATYPE command defines a unique two-character name that identifies a user data type and associates a Field exit routine with it.

The DataRefresher UIM records your CREATE DATATYPE definitions in the DataRefresher FDTLIB data set.

CREATE DXTPSB command

The CREATE DXTPSB command describes your IMS databases, segments, and fields to DataRefresher. Usually, you describe each IMS database (or each group of IMS databases if the DXTPSB contains multiple DXTPCB statements) to DataRefresher only once.

The CREATE DXTPSB command includes:

- One or more DXTPCB statements. The DXTPCB statement names the physical IMS database to be extracted and propagated.

- One or more SEGMENT statements. The SEGMENT statement names the physical segment to be extracted and propagated, as well as its physical parent and ancestors. The statement also indicates whether a Segment exit routine needs to be called.

  If you are using mapping case 3 propagation requests to propagate segments containing embedded structures, you also use one SEGMENT statement to describe each embedded structure. Embedded structures are called *internal segments* in this book.
- Multiple FIELD statements. The FIELD statement assigns a symbolic name to each field and describes the field in detail. For example, the statement describes the data format, length, and starting position of the field within the segment.

  If a segment is not processed by an IMS DPROP Segment exit routine, the definitions you provide in the FIELD statement should describe the fields of the segment as they appear in the I/O area of an IMS call.

  If a segment is processed by an IMS DPROP Segment exit routine, then the definitions you provide on the FIELD statements describe the fields in the edited format of the segment. The edited format for IMS-to-DB2 mapping is the segment format *after* editing by the Segment exit routine

  The edited format is also often called the IMS DPROP format. Field formats and field positions within the unedited segment format are transparent to IMS DPROP and DataRefresher.

  If the identified field format is a user data type, then during processing of the FIELD statement, DataRefresher UIM calls the Field exit routine identified on the CREATE DATATYPE command. This type of call to the Field exit routine is known as a definition (DEF) call. The definition call allows the Field exit routine to validate and complement the information provided on the FIELD statement.

DataRefresher UIM stores the definitions you provided on the CREATE DXTPSB command in the DataRefresher FDTLIB data set; therefore the definitions are available when DataRefresher processes your CREATE DXTVIEW and SUBMIT commands.

For generalized mapping cases, IMS DPROP does not support all options provided by DataRefresher on the DXTPCB, SEGMENT, and FIELD statements.

Related Reading: The *IMS DataPropagator for z/OS Reference* describes in detail which options are supported by IMS DPROP.

CREATE DXTVIEW command

Each CREATE DXTVIEW command describes one hierarchical path of the database from which IMS data is extracted and propagated. You can also use DXTVIEW commands to identify a subset of the IMS fields described in the CREATE DXTPSB.

Each CREATE DXTVIEW refers to a DXTPCB. on the CREATE DXTVIEW command, you identify which fields of one hierarchical path should be included in the view.

As described in the *IMS DataPropagator for z/OS Reference*, you need to provide at least one CREATE DXTVIEW command for each hierarchical path of the IMS database containing segments to be extracted and propagated. For propagation requests belonging to mapping case 2, you need to provide one CREATE

DXTVIEW command for each extension segment type. The DataRefresher UIM stores the definitions you provide in CREATE DXTVIEW commands in the FDTLIB data set for later reference.

SUBMIT command and EXTRACT statement

After creating the DATATYPEs, DXTPSBs, and DXTVIEWs, you can define propagation requests by providing one DataRefresher SUBMIT command with a DataRefresher EXTRACT statement for each propagation request to be defined. In DataRefresher, the propagation requests being defined are called extract requests.

The SUBMIT command assigns an eight-byte propagation request identifier (PR ID) to the propagation requests. The PR ID is also used as the name of the SQL update module that IMS DPROP generates whenever the propagation request, defined with MAPDIR=HR or TW, uses one of the generalized mapping cases.

The DataRefresher EXTRACT statement:

- Identifies the name of the DB2 table
- Refers to one or more DXTVIEWs
- Associates each IMS field to be propagated with a DB2 column
- Refers to only one DXTVIEW, for mapping case 1
- Refers to one DXTVIEW for each extension segment, for mapping case 2

To define propagation requests, you must specify a MAPEXIT=EKYMCE00 keyword on the DataRefresher SUBMIT command. The DataRefresher UIM then calls the IMS DPROP-provided Map Capture exit routine EKYMCE00. You should also provide IMS DPROP-specific information—such as the mapping case number and, for user mapping cases, the name of a Propagation exit routine—either on the MAPUPARM keyword of the SUBMIT statement or in a data set containing IMS DPROP default values. IMS DPROP-specific information is described in “Propagation parameters” on page 111.

The DataRefresher UIM provides to EKYMCE00 the data definitions and mapping definitions you specified on the CREATE DATATYPE, CREATE DXTPSB, CREATE DXTVIEW, and SUBMIT commands. When called by the DataRefresher UIM, IMS DPROP:

- Validates the information provided by the DataRefresher UIM. To validate, IMS DPROP needs DBD information from IMS DBDLIB and a table description from the DB2 catalog. The IMS DBD must be defined and the DB2 table must be created before IMS DPROP processes the propagation requests.

  For a propagation request belonging to a generalized mapping case, IMS DPROP determines which segment is the entity segment based on specifications you provided on the CREATE DXTVIEW commands and on fields that you identify in the EXTRACT statement.

  For a user mapping case, you must explicitly identify which segment types are propagated by the propagation request being defined. Specify the segment types in the PROPSEGM keyword of MAPUPARM or of an MVGPARM default data set. For IMS-to-DB2 propagation, IMS DPROP calls the Propagation exit routine associated with the propagation request each time one of these segment types is updated.
- Creates a propagation request in the mapping tables of the IMS DPROP directory if validation is successful. The propagation request contains data definitions and mapping definitions provided by DataRefresher UIM and additional information gathered by IMS DPROP from DBDLIB and the DB2 catalog. For each propagation request, IMS DPROP stores information into the

mapping tables of the IMS DPROP directory. This information is described in Chapter 4, “Control information and environment,” on page 73.

As you run the submit command or immediately after running the command, a series of events can occur:

- If warning messages are generated when the propagation request is created, the messages are recorded in the MSG table in the IMS DPROP directory and written to a print file.
- Flags in the IMS DPROP directory are set to indicate that the status of the propagation request just created is *inactive.*
- One RUP propagation request control block (PRCB) is created for each propagated IMS segment. The control block contains mapping information for all propagation requests propagating from or to a particular segment. Mapping information includes the displacement, length, and format of the fields to be propagated, as well as the DB2 table name, mapping case, and error option. For performance reasons, RUP PRCBs are located in both the IMS DPROP directory and the Virtual Lookaside Facility (VLF) of MVS/ESA.
- MVG updates the master timestamp field in the master table in the IMS DPROP directory to signal changes to RUP and HUP. If RUP or HUP detect changes to the directory, they refresh directory objects stored in memory so the changes become effective. The unique row of the master table is also stored in VLF to improve performance.
- For propagation requests belonging to a generalized mapping case and specifying MAPDIR=HR or TW, MVG also generates the assembler source code for an SQL update module. The module contains all the SQL update statements required to propagate data from IMS to DB2 based on the propagation request definitions. The SQL source is pre-compiled, assembled, and linked into a load library as an SQL update module by IMS DPROP. You must then use a DB2 BIND operation to bind the DBRM of the SQL update module into either a DB2 package or the plans of propagating applications. You must do the bind before the DBRM can be used in propagation. You might want to bind the DB2 package automatically by using MVG.

If the propagation request is created without errors, IMS DPROP returns to the calling DataRefresher UIM. UIM then stores the corresponding extract request in the DataRefresher EXTLIB data set. The definitions are then available in EXTLIB to the DataRefresher DEM when an extract is done.

You should save your DataRefresher SUBMIT and EXTRACT specifications, because you need to provide them to DataRefresher every time you want to extract IMS data with DataRefresher. After successful completion of the extract, the DEM deletes the extract request from EXTLIB. However, IMS DPROP keeps the propagation request definitions in the IMS DPROP directory until you delete them using the IMS DPROP MVGU.

You can use DataRefresher to build the propagation request, without extracting IMS data with the DataRefresher DEM. To do so, specify PERFORM(BUILDONLY) in the MAPUPARM keyword of the DataRefresher EXTRACT statement.

If you use the DataRefresher UIM to define propagation requests, both IMS DPROP and DB2 functions are called. UIM JCL needs to be modified with JCL required by IMS DPROP and DB2. The propagation request definition process using DataRefresher is illustrated in Figure 26 on page 106.

Related Reading: Refer to the *IMS DataPropagator for z/OS Reference* for more detailed information and examples.

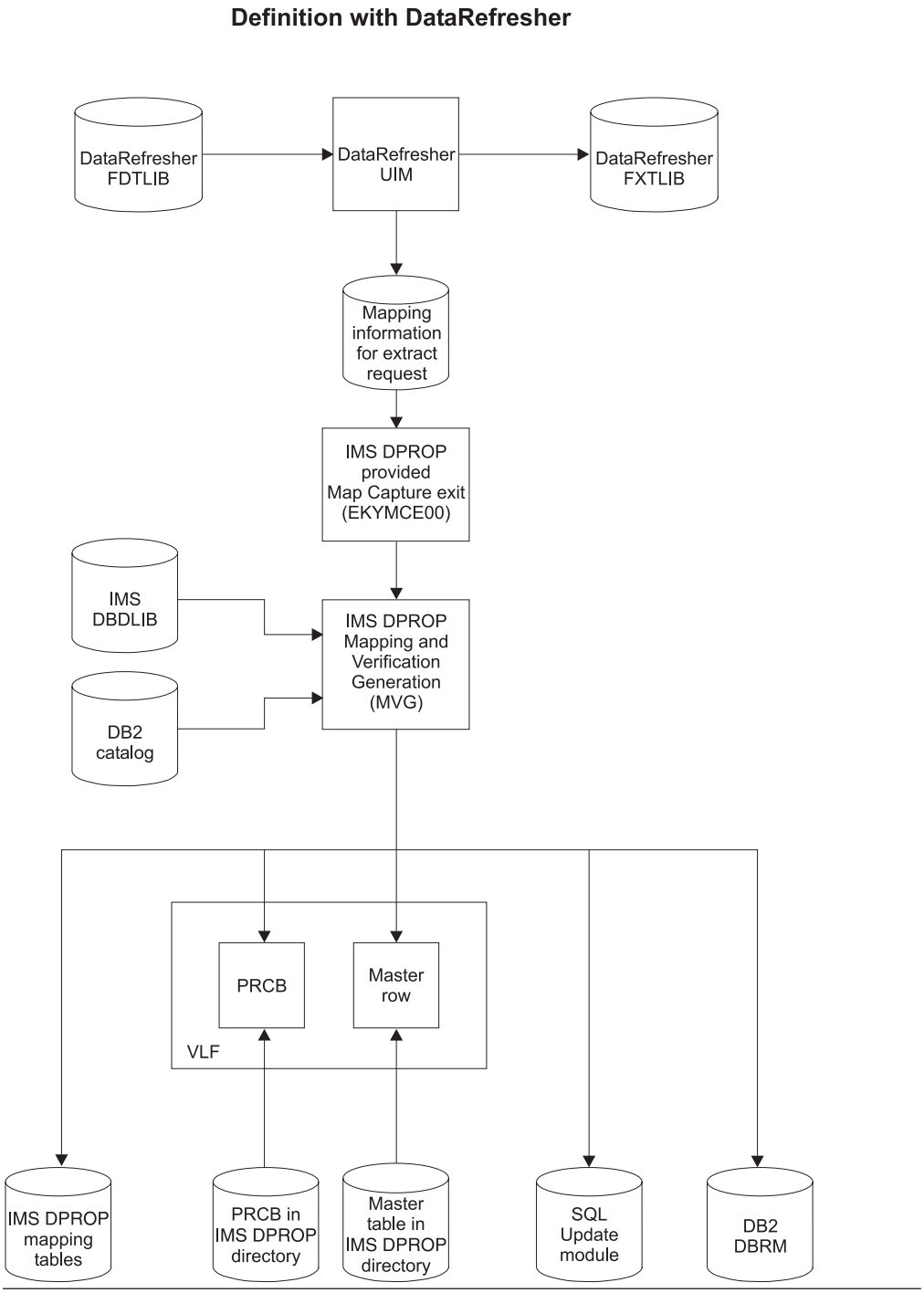


Figure 26. Propagation request definition with DataRefresher

DataRefresher and user mapping cases

For some segments, your mapping requirements might not be satisfied by the generalized mapping logic of IMS DPROP. Usually, such segments are propagated with Propagation exit routines. Propagation exit routines perform mapping, data

conversions, and SQL updates during IMS-to-DB2 propagation and IMS updates during DB2-to-IMS synchronous propagation.

For user mapping cases, you might consider using the mapping and conversion capabilities of DataRefresher for extracts so that you don't have to write extract programs. The mapping and conversion done by DataRefresher should be compatible with the mapping and conversion done by your Propagation exit routine. Otherwise, you need to provide your own extract programs.

When determining whether to use DataRefresher to extract propagation requests belonging to a user mapping case, be aware that DataRefresher supports the following mapping capabilities:

- Nesting of internal segments and repeating groups of fields, so that an internal segment can contain, in turn, other internal segments
- Joining data from multiple IMS databases

If you need more information about the mapping capabilities of DataRefresher, or DXT refer to the appropriate library. Sample DataRefresher definitions for Propagation exit routines are discussed in the *IMS DataPropagator for z/OS Reference*.

Defining propagation requests using the MVG input tables

IMS DPROP provides an ISPF/TSO interface that you can use to define, update, and delete propagation requests stored in the MVG input tables. However, this section assumes you are not using the ISPF/TSO interface but instead are using SQL statements to build propagation requests in the MVG input tables.

The five MVG input tables are:
- PR table—propagation request table (DPRIPR)
- TAB table—target DB2 table (DPRITAB)
- SEG table—IMS segment table (DPRISEG)
- FLD table—IMS field table (DPRIFLD)
- WHR table—WHERE table (DPRIWHR), containing the WHERE clause of the propagation request

Related Reading: For more details on the MVG input tables and their columns, refer to the *IMS DataPropagator for z/OS Reference*.

To define a propagation request for a generalized mapping case, you must provide a row in DPRIPR, one or more rows in DPRISEG, one row in DPRITAB, and one or more rows in DPRIFLD. If defining a propagation request with a WHERE clause, you must also provide one or more rows in the DPRIWHR table.

To define a propagation request for user mapping, you must provide at least one row in DPRIPR, DPRITAB, and DPRISEG.

This section covers:
- Identifying the propagation request
- Specifying the IMS segments to be propagated
- Specifying the DB2 tables
- Specifying the fields
- Executing the MVGU

Identifying the propagation request

DPRIPR contains one row of information for each propagation request being defined. Information includes the PR ID, which is also stored in all other tables of the MVG input tables. Storing the PR ID in all MVG input tables lets rows of the MVG input tables be identified as belonging to a specific propagation request. The PR ID is also used as the name of the SQL update module that IMS DPROP generates whenever the propagation request is defined with MAPDIR=HR or TW and uses one of the generalized mapping cases.

Specifying the IMS segments to be propagated

You must identify the segments to be propagated by the propagation request being defined. If you are using generalized mapping cases, DPRISEG must contain a row for the entity segment and for each physical ancestor of the segment to be propagated, up to and including the root segment. If you are using mapping case 2, additional rows must exist for extension segments.

If you are using mapping case 3 and propagating an embedded structure, DPRISEG must also contain one row for each structure embedded in the segment. Embedded structures are referred to as internal segments.

If you are using a Propagation exit routine for user mapping, DPRISEG must contain a row for each segment to be propagated by the propagation request being defined. At IMS-to-DB2 propagation time, IMS DPROP calls the Propagation exit routine every time one of the segments is updated.

Specifying the DB2 tables

DPRITAB must contain one row for each DB2 table propagated by the propagation request. Generalized mapping allows only a single propagated table for each propagation request. A propagated table can have either a fully qualified or unqualified name.

Specifying the fields

DPRIFLD must contain one row for each propagated field defined in the propagation requests. A given field in a segment can be either selected or not selected for propagation. If a field is selected for propagation, it must have a corresponding target column. For a non-selected field, the column name is left blank.

Running the MVGU

Once you have provided the necessary propagation request information in the MVG input tables, you should run the MVG utility (MVGU). MVGU retrieves the mapping information from the MVG input tables, constructs a control block from the information retrieved, and calls MVG. MVG validates the information stored in this control block, such as propagation request type and propagation mode.

If you are using a generalized mapping case, MVG extracts the following information from the IMS DBD defining the database being propagated:

- Parent segment name
- Segment length
- Segment key field name
- Segment key field length
- Segment key field offset
- Segment format

- Database organization

Similar information for the target DB2 tables is taken from the DB2 catalog. The target or model DB2 tables must, therefore, be created in the DB2 system before MVG processes the propagation request.

If the propagation request information is successfully validated by MVG, then MVG stores the mapping information in the mapping tables in the IMS DPROP directory. Flags in the IMS DPROP directory indicate that the status of the propagation request just created is *inactive.*

When you run or immediately after you run MVGU, a series of events can occur:

- A RUP PRCB is created, one for each propagated IMS segment. It contains mapping information for all propagation requests propagating from or to a particular segment. Mapping information includes the displacement, length, and format of the IMS fields to be propagated, as well as the DB2 table name, mapping case, and error option. For performance reasons, RUP PRCBs are located in both the IMS DPROP directory and in VLF.
- MVG updates the master timestamp field in the master table in the IMS DPROP directory to signal changes to RUP. If RUP detects changes to the directory, it refreshes directory objects that are stored in memory so the changes become effective. The unique row of the master table is also stored in VLF to improve performance.
- For propagation requests belonging to a generalized mapping case and specifying MAPDIR=HR or TW, MVG also generates the assembler source code for an SQL update module. This module contains all the SQL update statements required to propagate data from IMS to DB2 based on propagation request definitions. IMS DPROP pre-compiles, assembles, and links the SQL source into a load library as an SQL update module. You must then bind the DBRM of the SQL update module either into a DB2 package or the plans of propagating applications before it can be used in propagation. Use a DB2 BIND operation to bind. You might want to bind the DB2 package automatically by using MVG.
- A flag in the propagation request in the MVG input table is set to indicate that the MVGU has processed the propagation request and placed it in the IMS DPROP directory.

The propagation request definition process using the MVG input tables is illustrated in Figure 27 on page 110.

PR Definition with MVG Input Tables

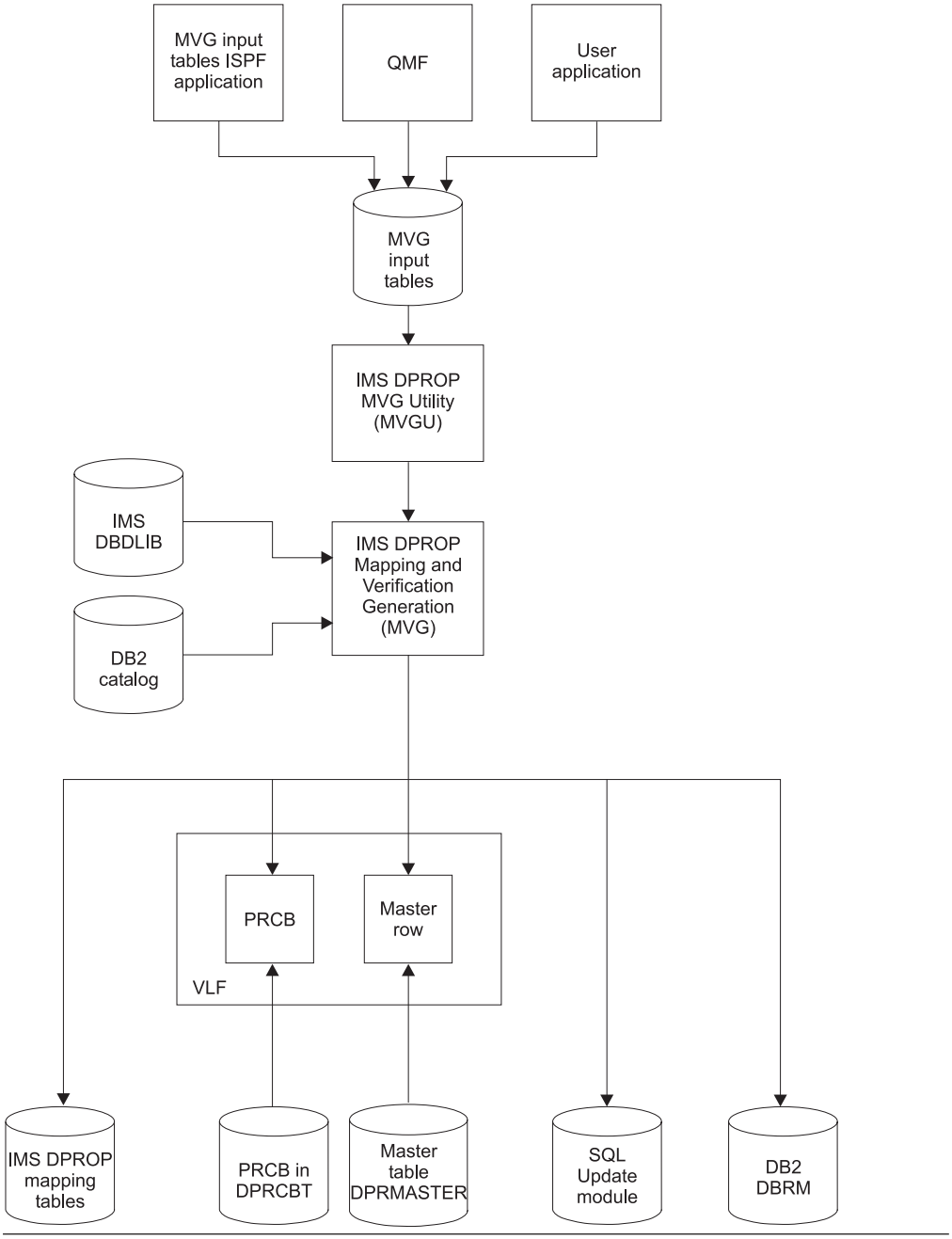


Figure 27. Propagation request definition with MVG input tables

Propagation parameters

You must specify certain parameters when you build a propagation request. If you are using DataRefresher, the parameters are specified in the MAPUPARM parameter of the DataRefresher SUBMIT command. If you are using the MVG input tables to define propagation requests, the parameters are specified in DPRIPR. You can provide default values for propagation parameters in the //MVGPARM data set. Propagation parameters specify:

- Propagation request type
- Mapping case
- PATH data option
- Mapping direction
- DB2 table qualifier used for validation
- Error option
- Maximum number of error messages
- PR action
- Propagation request set name
- Propagation suppression
- Whether to avoid unnecessary updates
- Ordering sequence of the DB2 primary key
- Type of operation, used only with DataRefresher
- Exit name
- Propagated segments for user mapping, used only with DataRefresher
- Package bind options

This section briefly describes each parameter. For detailed information on these parameters and how to specify them, refer to the *IMS DataPropagator for z/OS Reference*.

For additional considerations for LOG-ASYNC propagation, see “Changing propagation requests or propagation groups” on page 195.

PRTYPE—type of propagation request

Specifies the propagation request type to be created. Valid propagation request types are:

E Extended function
L Limited function
U User mapping
F Full function. Used only for compatibility with IMS DPROP R1

MAPCASE—Mapping case

Specifies the mapping case. The generalized mapping cases are 1, 2 and 3. You do not have to specify a mapping case for user mapping PRTYPE=U.

PATH—Path data option

Specifies whether path data is included in the mapping of a generalized mapping case.

Specify PATH=ID if no fields included in the path data change their values.

Specify PATH=DENORM if some fields included in the path data can change their values. PATH=DENORM usually results in denormalization of data.
PATH=DENORM is not valid for PRTYPE=E propagation requests.

MAPDIR—Mapping direction

Specifies the propagation direction as follows:

HR Hierarchical to relational only. For one-way IMS-to-DB2 propagation only.

TABQUAL2—DB2 table qualifier used for validation

Specifies a propagation request with unqualified table names.

Propagation requests can be defined with qualified or unqualified table names for the propagated table. Propagation requests with qualified table names are usually used in production environments. They support propagation to or from only one table whose qualified name is defined in the propagation request.

Unqualified table names can be defined in propagation requests for some test environments. They can support propagation to or from one of multiple, identically structured tables. For IMS-to-DB2 propagation, each table must have its own plan bound for the propagating application. If more than one propagation request is to propagate to the same table, each of the propagation requests should be bound into a different plan that specifies a unique table identifier. Support for multiple copies also requires that propagated IMS DBDs have the same name and that DB2 tables have the same unqualified name.

If you define a propagation request with unqualified table names, you specify a qualifier on the TABQUAL2 parameter. MVG uses the qualifier to identify a *model table* in the DB2 catalog. MVG generates mapping information in the propagation request based on the DB2 catalog description of the model table. Therefore, the model table should have the same attributes as the propagated tables.

For more information on defining propagation requests with qualified or unqualified table names, refer to “Defining propagation requests with qualified or unqualified table names” on page 44.

ERROPT—error option

Specifies the error option (BACKOUT or IGNORE) to be taken when a propagation request fails.

MAXERROR—Maximum number of reported propagation errors

Specifies how many propagation failures for a propagation request are to be reported to the console and the audit trail. This parameter applies only when ERROPT=IGNORE is specified.

ACTION

Specifies whether the propagation request is to be added or replaced.

PRSET—Propagation request set name

Specifies the set of propagation requests (PRSET) to which a single propagation request belongs. For more information, refer to the *IMS DataPropagator for z/OS Reference*.

PROPSUP—Propagation suppression

Specifies whether RUP should accept or reject a return code of 8 from a Segment exit routine. The Segment exit routine can use a return code of 8 to request suppression of propagation. For IMS-to-DB2 propagation, it is recommended that you suppress propagation by defining a WHERE clause during propagation request definition, when possible.

Related Reading: For more information, refer to the *IMS DataPropagator for z/OS Reference*.

AVU—Avoid Unnecessary Updates

Specifies whether IMS DPROP, when replacing a segment, should determine if at least one propagated field has changed. The parameter lets you influence the performance of IMS-to-DB2 propagation.

If you set AVU to YES, a replaced IMS segment results in a propagating SQL update only if at least one propagated field has changed. IMS DPROP compares the before-image and after-image of the replaced segment to determine whether any propagated field has changed. The path length is increased for the comparison, especially if the mapping involves a Segment exit routine, which is be called twice. But you avoid issuing unnecessary SQL update statements when no propagated field has changed. AVU set to YES can reduce the total IMS DPROP path length when only a subset of segment fields are propagated.

When AVU is set to NO, a replaced IMS segment results in a propagating SQL update even if no propagated field has changed. IMS DPROP does not compare the before- and after-image of the changed segment to determine whether any propagated field has changed. You do not require increased path length for the comparison. And it can also reduce total IMS DPROP path length when, for example, all fields in a segment are propagated.

You usually do not need to provide an AVU parameter.

IMS DPROP uses AVU=Y when processing changes of either:

- Internal segments propagated by mapping case 3 propagation requests
- IMS segments propagated by mapping case 1 or 2 propagation requests when at least one non-key byte of the segment is not propagated.

IMS DPROP uses AVU=N in all other cases.

You might want to override the IMS DPROP default value and specify AVU=N for mapping case 1 and 2 when either:

- Most IMS replace operations will change at least one propagated field
- Your Segment exit routines have a large path length

KEYORDER—DB2 key ordering sequence

Specifies whether the columns of the DB2 primary key of the propagated table are ordered by the primary key index in ascending or descending sequence or whether MVG must access the DB2 catalog to determine the ordering sequence of each column. KEYORDER applies to all columns used in the primary key. If you have both ascending and descending columns used in the key, you must specify KEYORDER=ANY. Depending on what you specify you might have lengthy accesses to the DB2 catalog to determine the key order of each column.

PERFORM—Type of operation: DataRefresher only

Specifies whether IMS DPROP and DataRefresher are to:

- Create a propagation request and store the extract request in EXTLIB (BUILDRUN)
- Create a propagation request without storing the extract request (BUILDONLY)
- Store the extract request only (RUNONLY)

You can run extract requests stored in EXTLIB using the DataRefresher DEM.

EXITNAME—Name of Propagation exit

When you propagate using a Propagation exit routine (PRTYPE=U), specifies the name of the exit routine.

PROPSEGM—Propagated segments: User mapping with DataRefresher only

Identifies the segments that are to be propagated by the propagation request being defined. At IMS-to-DB2 propagation time, the Propagation exit routine is called when one of the identified IMS segments is updated.

For DB2-to-IMS synchronous propagation, IMS DPROP calls the Propagation exit routine for the propagation request each time a propagated table identified in the DataRefresher EXTRACT statement or in the DPRITAB table is updated.

BIND—Options for a DB2 package bind

It is recommended that you use the DB2 package bind function. Binding the DBRM of the SQL update modules of your propagation requests can simplify administration of your propagating application program's plans.

If you provide a BIND parameter, MVG automatically binds the DBRM of your SQL update modules into a DB2 package. Specify in the BIND parameter the options MVG should use for package binding the SQL update module. Among other things, specify the collection ID where the package will be bound.

Deleting a propagation request

To delete a propagation request from the IMS DPROP directory and delete the SQL update module from the load library and DBRM library, you must run MVGU with DELETE control statements. The DELETE control statement can also delete the DB2 package of the SQL update module for the propagation request.

Do not use SQL deletes to delete propagation requests from the IMS DPROP directory tables. You create inconsistencies in IMS DPROP's control information, leading to unpredictable errors and jeopardizing data propagation.

In the MVGU DELETE statement, you can specify one or more:

- Propagation requests to delete
- Segment names—to delete propagation requests propagating the segment types
- Databases—to delete propagation requests propagating the databases

When processing a DELETE, the MVGU does not access the MVG input tables. MVGU deletes only specified propagation requests from the IMS DPROP directory. To delete information about a propagation request from the MVG input tables, you must use the SQL DELETE statement.

DataRefresher UIM does not call IMS DPROP when DataRefresher CANCEL commands are processed. DataRefresher users must use MVGU to delete propagation requests.

Related Reading: For detailed information on how to code the DELETE command, see the *IMS DataPropagator for z/OS Reference*.

Replacing a propagation request

To change a propagation request in the IMS DPROP directory, the propagation request must be recreated by using either DataRefresher or the MVG input tables in the same process used to create the original propagation request. SQL statements should *not* be used to change a propagation request in the IMS DPROP directory.

Ensure that the ACTION propagation parameter in the MVGPARM parameter is specified as REPL. Also, set the propagation request to INACTIVE state by running the SCU:

- If you are using DataRefresher to recreate the propagation request, change the DataRefresher statements as desired and then rerun DataRefresher UIM. If you want to change the propagation request without creating an extract request, use the propagation parameter PERFORM=BUILDONLY.
- If you are using the MVG input tables, use SQL to change input table information. You must set the PROCSED column of DPRIPR to either blank or N to indicate that the changed propagation request should be processed. Then you can issue the MVGU CREATE statement to update a propagation request in the IMS DPROP directory.

Related Reading: For additional information on changing a propagation request, see the *IMS DataPropagator for z/OS Reference*.

Rebuilding a propagation request

You can use the MVGU RECREATE function to rebuild propagation control blocks in the IMS DPROP directory. The RECREATE function can also rebuild SQL update modules if they are destroyed. MVG retrieves information from the mapping tables and uses the information to recreate the objects for:

- Specific propagation requests
- Propagation requests propagating specific segments or databases
- All propagation requests defined in the mapping tables of the IMS DPROP directory

When processing RECREATE, MVGU does not access the MVG input tables, only the IMS DPROP directory. The RECREATE function does *not* alter the propagation requests stored in the mapping tables of the IMS DPROP directory.

For detailed information on how to code the RECREATE control statement, see the *IMS DataPropagator for z/OS Reference*.You might also want to refer to “Changing propagation requests or propagation groups” on page 195

Revalidating propagation requests

The MVGU utility has a REVALIDATE function. MVGU revalidation is used to revalidate propagation requests that have been defined earlier. Use revalidation ensure propagation request definitions are still valid after possible changes to IMS database definitions or DB2 table definitions.

You can also use MVGU revalidation to verify that DB2 RIRs are compatible with physical and logical IMS parent/child relationships. The referential integrity checking done by MVGU revalidation is usually more complete than the checking done when the propagation request is defined because you can run it after all propagation requests have been defined. The RIR checking done by MVGU revalidation considers the “whole picture.”

MVGU revalidation is usually run after a set of PR definitions are completed, after definitional changes to IMS and DB2, and on a periodic basis.

Chapter 7. Granting privileges and authorizations for DB2 objects

This chapter discusses DB2 privileges in a data propagation environment. You must grant DB2 privileges or authority for different types of objects. This chapter distinguishes between granting privileges for:

- IMS DPROP tables, IMS DPROP utilities, and related objects
- Your propagated tables, your propagating applications, and associated objects

For IMS DPROP tables, utilities, and related objects, this chapter describes:

- Granting privileges for IMS DPROP directory tables, the audit trail table, and the MVG input tables
- If you use the DB2 package bind facility, binding the packages of IMS DPROP modules accessing IMS DPROP tables
- If you use the DB2 package bind facility, granting privileges for the two collection IDs containing the packages of:
  - IMS DPROP modules reading IMS DPROP tables
  - IMS DPROP utility modules updating IMS DPROP tables
- Binding the DB2 plans of IMS DPROP utilities
- Running IMS DPROP utilities

For your propagated tables, propagating applications, and related objects, this chapter describes:

- Granting privileges for your propagated tables
- If you use the DB2 package bind facility, granting privileges for the collection IDs containing the packages of SQL update modules and exit routines updating your propagated tables
- If you use the DB2 package bind facility, binding packages of SQL update modules and exit routines accessing the propagated tables
- Binding DB2 plans of propagating application programs
- Running propagating application programs

DB2 security mechanisms are very flexible, so you can establish DB2 privileges and authority many ways. This chapter provides general recommendations and describes only one of the many ways to establish DB2 security for data propagation.

To understand this chapter, you need to be familiar with DB2 security mechanisms. For more information on DB2 security, see *DB2 Administration Guide*.

IMS DPROP tables, utilities, and related objects

This section describes privileges and authority related to IMS DPROP tables, utilities, and related objects. Topics included in this section are:

- Granting privileges for IMS DPROP tables
- Binding packages of IMS DPROP modules
- Granting privileges for IMS DPROP collections
- Binding plans of IMS DPROP utilities
- Granting privileges for running IMS DPROP utilities

Granting privileges for IMS DPROP tables

You need to secure the following DB2 tables:
- IMS DPROP directory tables
- MVG input tables
- Audit trail table

IMS DPROP directory tables

Generally, the only people who should have privileges granted to them beyond SELECT for the IMS DPROP directory are those who own DB2 packages or DB2 plans used by IMS DPROP utilities. This prevents inadvertent updates to the IMS DPROP directory tables.

You can grant the SELECT privilege to PUBLIC for the following tables:
- DPRMASTER
- DPRCBT
- DPRHCBT
- DPRPR
- DPRWHR
- DPRMSG
- DPRSEG
- DPRTAB
- DPRFLD
- DPRRCT
- DPRPRCT
- DPRPRDSR
- DPRDRDSV

You should only update the directory tables with IMS DPROP utilities, MVG, and SCU. Do not use your own applications or QMF to insert, update, or delete rows in these tables. If you do so, the tables can contain erroneous or inconsistent control blocks, and IMS DPROP could generate unpredictable results.

MVG input tables

If you define all your propagation requests using DataRefresher, then you do not need to grant any privileges or even build the MVG input tables.

If you are using the MVG input tables to build propagation requests, then you need to grant the SELECT, UPDATE, INSERT, and DELETE privileges to the authorization identifiers used by people who:
- Build propagation requests in the MVG input tables
- Own the DB2 packages or plan of the MVG

The MVG input tables include:
- DPRIPR—propagation request table (PR table)
- DPRIWHR—WHERE clause table (WHR table)
- DPRITAB—target DB2 table (TAB table)
- DPRISEG—IMS segment table (SEG table)
- DPRIFLD—IMS field table) (FLD table)

Audit trail table

The audit trail table has three levels of privileges:
- SELECT privileges for people querying the audit trail table
- SELECT, UPDATE, INSERT, and DELETE privileges for people maintaining the audit trail table (for example, deleting old or outdated rows)

- SELECT, UPDATE, INSERT, and DELETE privileges for people owning the packages or plan of the AUDU utility

Binding packages of IMS DPROP modules

During IMS DPROP installation, you specify whether you intend to use the DB2 package bind facility. If using the facility, you identify two collection IDs for each IMS DPROP system. The collection IDs are referred to as the “IMS DPROP collections.”

The IMS DPROP installation process binds the packages of IMS DPROP modules into these two collection IDs.

- The first IMS DPROP collection is used to bind packages of IMS DPROP utility modules reading and updating the IMS DPROP directory tables. It is called the “read-write IMS DPROP collection.”
- The second IMS DPROP collection is used to bind packages of IMS DPROP modules reading IMS DPROP directory tables. It is called the “read-only IMS DPROP collection.”

The authorization ID you use to do IMS DPROP installation must have the following privileges:

- BINDADD and CREATE IN COLLECTION privilege, for binding new packages, or BIND privilege, if binding again an existing package
- SELECT, UPDATE, INSERT, and DELETE privileges for the IMS DPROP directory tables, MVG input tables, and audit trail table
- SELECT privilege for the DB2 catalog tables, needed because some IMS DPROP modules read information from the DB2 catalog

Binding plans using package bind are further discussed in Chapter 8, “Binding and administering plans,” on page 127.

Granting privileges for IMS DPROP collections

As part of the installation process, you are also asked to grant the CREATE IN COLLECTION and EXECUTE privileges for the IMS DPROP two collections. The authorization ID you use to do IMS DPROP installation must have the authority to grant privileges for the two IMS DPROP collections. See “Binding packages of IMS DPROP modules” on page 119.

When granting the privileges:

- Be restrictive when granting the CREATE IN COLLECTION privilege. Usually, only the IMS DPROP system administrator needs to bind into these collections packages of IMS DPROP modules. Therefore, only an authorization ID used by the system administrator needs these privileges for these collections.
- Be restrictive when granting the EXECUTE privilege for the read-write IMS DPROP collection. Usually only the IMS DPROP system administrator needs to bind and own the plans of IMS DPROP utilities. Therefore, only an authorization ID used by the system administrator needs the EXECUTE privilege for these collections.
- You do not need to be restrictive when granting the EXECUTE privilege for the read-only IMS DPROP collection. All owners of DB2 plans of propagating applications and IMS DPROP utilities need the EXECUTE privilege. Since the packages of this collection provide read-only access, you might want to grant the EXECUTE privilege to PUBLIC.

Also consider granting BIND and COPY privileges for the two IMS DPROP collections. Be restrictive when granting these privileges. Usually only IMS DPROP system administrators need these privileges.

Binding plans of IMS DPROP utilities

During IMS DPROP installation, you should bind the DB2 plans of the following IMS DPROP utilities:

- AUDU
- CCU
- MVGU
- SCU
- DLU

If you use the DB2 package bind facility, binding these plans requires the following authorizations:

- BINDADD privilege, for binding new plans, or BIND privilege, if binding again an existing plan
- EXECUTE privilege for the DB2 collection IDs containing IMS DPROP packages

If you do not use the DB2 package bind facility, binding the IMS DPROP utility plans requires the following authorizations:

- BINDADD privilege, for binding new plans, or BIND privilege, if binding again an existing plan
- SELECT, UPDATE, INSERT, and DELETE privileges for the IMS DPROP directory tables, MVG input tables, and audit trail table
- SELECT privilege for DB2 catalog tables needed because some IMS DPROP modules read information from the DB2 catalog

After binding the plans for the utilities, you need to grant the EXECUTE privilege for them. Usually, this privilege is granted to authorization IDs used by systems programmers, database administrators, and operations personnel.

Binding plans using package bind are further discussed in Chapter 8, “Binding and administering plans,” on page 127.

Running IMS DPROP utilities

Running an IMS DPROP utility requires the EXECUTE privilege for the DB2 plan of the utility. Execution of some IMS DPROP utilities requires *additional* privileges, as described in this section:

- Additional authorizations required to execute CCU
- Additional authorizations required to run MVG/MVGU
- Additional privileges required to execute the SCU
- Additional authorizations required to execute the IMS DPROP utilities front end applications

Additional authorizations required to run the CCU

The CCU reads the rows of the propagated tables with dynamic SQL statements. Therefore, the person who runs the CCU needs the SELECT privilege for the propagated tables.

Additional authorizations required to run MVG/MVGU

When creating or recreating propagation requests for a generalized mapping case for IMS-to-DB2 propagation, MVG creates an SQL update module. As an option, MVG does an automatic package bind of the DBRM of the SQL update module into the collection ID that you specify.

To use the MVG bind option, the person who runs the MVG must have either the SYSADM or the SYSCTRL privilege or must be granted all the following privileges:

- BINDADD and CREATE IN COLLECTION privilege for the collection ID where the package of the SQL update module is bound, for binding new packages, or BIND privilege for the package, if re-binding an existing package.
- SELECT, UPDATE, INSERT, and DELETE privilege on the DB2 propagated table affected by the propagation request. These privileges are also necessary for people having the SYSCTRL privilege.

When deleting a propagation request, MVGU can delete packages previously bound by MVG. For MVGU to delete a package, you must own the package, have the package owner grant you the BINDAGENT privilege, or you must be granted SYSCTRL or SYSADM authority.

Additional privileges required to execute the SCU

Various DB2 privileges must be granted to execute the following SCU control statements:

- READON and READOFF control statements for DB2 databases and table spaces. When processing these control statements, the SCU issues internally DB2 START DATABASE and DISPLAY DATABASE commands. Therefore, the person who executes the SCU must be granted the STARTDB and DISPLAY privileges.

Additional authorizations required to run the IMS DPROP utilities front end applications

The CCU, DLU, and MVGIN front end applications read the rows of the IMS DPROP directory tables or MVG input tables with dynamic SQL statements. Therefore, the person who runs any of these front end applications needs the SELECT privilege for the IMS DPROP directory tables and for the MVG input tables.

Propagated tables, propagating applications, and related objects

This section describes privileges and authorization relating to:

- Propagated tables
- Propagating collections
- Binding packages of SQL update modules and Propagation exit routines
- SQL update modules bound into different packages
- DB2 plans of propagating applications
- Propagating applications

Granting table privileges for propagated tables

This section provides considerations for granting privileges for IMS-to-DB2 propagation.

When doing IMS-to-DB2 propagation, be restrictive when granting privileges beyond SELECT for propagated tables. This prevents updates to DB2 tables, which can result in inconsistencies between IMS and DB2 data. Grant table privileges other than SELECT only to authorization IDs that:

- Own the DB2 packages of SQL update modules or Propagation exit routines, if doing IMS-to-DB2 propagation with the DB2 package bind facility
- Own the DB2 plans of propagating applications, if you do IMS-to-DB2 propagation without the DB2 package bind facility
- Execute table repair programs, such as applying CCU-generated repair files, using the DB2-supplied programs DSNTEP2 or DSNTIAD
- Execute programs resynchronizing the DB2 copy after propagation has been suspended

You should grant the SELECT privilege for propagated tables to people:
- Using decision support systems
- Querying propagated tables
- Running CCU
- Running DLU

Updates to non-propagated columns

You can update non-propagated columns of propagated tables without causing inconsistencies between IMS and DB2. But, it is important to access propagated columns in read-only mode. If some columns of a propagated table are not to be propagated, those columns should either be defined as NOT NULL WITH DEFAULT or be defined to permit null values when the table is created.

To update non-propagated columns, use views containing those columns so that you can grant update authority to the view containing the columns, without granting authority at the table level.

Only update authority should be granted. The use of insert or delete authority jeopardizes data consistency. Inserts and deletes operate at the row level, while updates affect columns.

Figure 28 on page 123 illustrates the concept of updating non-propagated columns.

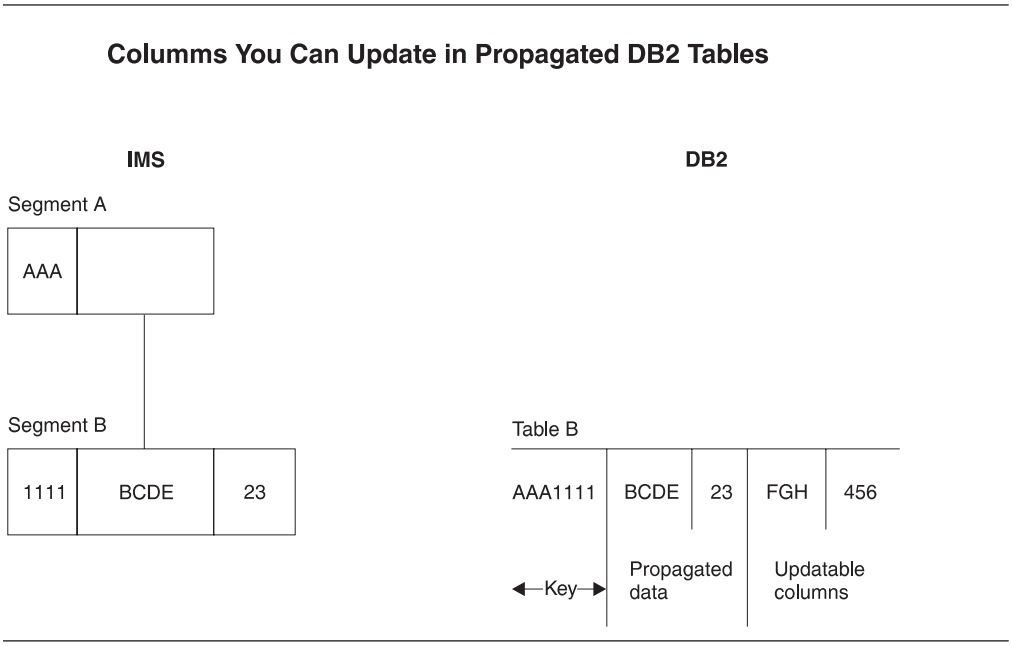


Figure 28. Columns that can be updated in propagated DB2 tables. The columns containing FGH and 456 can be updated through a view. The key and propagated data should be read-only.

If your IMS and DB2 data become inconsistent, you might not be able to resynchronize data using a re-extract or a CCU repair file, because some of your columns are non-propagated. You might have to provide a program of your own to resynchronize data.

Granting privileges for propagating collections

If you want to use the DB2 package bind facility and do IMS-to-DB2 propagation, you need to bind the DBRMs of SQL update modules and Propagation exit routines into DB2 packages. You need to decide into which collections the packages should be bound. These collections are referred to as the “propagating collections.”

You might want to use different propagating collections for production work and tests.

Consider the following DB2 privileges when propagating collections:

- The CREATE IN COLLECTION privilege must be granted to owners of the packages of SQL update modules and Propagation exit routines. This privilege is required for binding the packages.
- Consider granting the EXECUTE privilege for the *entire* collection ID, instead of individual packages, to the owners of plans of propagating applications. The EXECUTE privilege is required to bind the plans of propagating applications.

  If you do not grant the EXECUTE privilege for the entire collection ID to future owners of plans of propagating applications, you must grant the EXECUTE privilege for individual packages.

  Granting the EXECUTE privilege for the *entire* collection simplifies administration of DB2 plans of your propagating applications. It allows you, when binding the plans of propagating applications, to specify PKLIST(collection.*) instead of explicitly identifying each required package. You do not need to know which package is required for which plan.

- You also need to decide if you should grant the BIND and COPY privileges for the entire collection ID or for individual packages.

Binding packages of SQL update modules and Propagation exit routines

If you want to use the DB2 package bind facility and do IMS-to-DB2 propagation, you bind the DBRMs of SQL update modules and Propagation exit routines into DB2 packages. You can bind the DBRMs of SQL update modules as part of MVG processing when you create or recreate the propagation request.

The bind process requires that the owner of the DB2 package have the following privileges:
- BINDADD and CREATE IN COLLECTION privilege, for binding new packages, or BIND privilege for the package, if binding an existing package again
- CREATE IN privilege for the collection ID
- SELECT, UPDATE, INSERT, and DELETE privileges for the propagated tables

If you do not grant the EXECUTE privilege for the entire collection ID to future owners of plans of propagating applications, you must grant the EXECUTE privilege for individual packages. You might also need to grant the BIND and COPY privilege for individual packages.

Chapter 8, “Binding and administering plans,” on page 127 has additional information about binding packages.

Binding SQL update modules into different packages

If you have defined your propagation requests with unqualified table names, then you will often want to bind the DBRM of the SQL update module into different packages using different table-name qualifiers. You can do this using the COPY and QUALIFIER keywords of the DB2 BIND command.

A BIND with the COPY option uses a previously-bound package of the SQL update module as input and creates a new package accessing the propagated tables with the specified QUALIFIER keyword.

The BIND COPY process requires that the owner of the new package have the following privileges:
- COPY privilege for the package, or its collection, being copied
- BINDADD privilege and BIND privilege for the package or collection
- CREATE IN privilege for the collection ID
- SELECT, UPDATE, INSERT, and DELETE privileges for the propagated tables

If you do not grant the EXECUTE privilege for the entire collection ID to future owners of plans of propagating applications, you must grant the EXECUTE privilege for individual packages.

Chapter 8, “Binding and administering plans,” on page 127 has additional information about binding packages.

Binding DB2 plans of propagating applications

You must bind DB2 plans for propagating applications and for the receiver programs used in user asynchronous propagation. This requirement stems not only from the SQL updates made by IMS DPROP, but also from SQL reads of the IMS DPROP directory, which is composed of DB2 tables.

To bind the plans, the plan owner must have the following privileges if using the DB2 package bind facility:

- BINDADD privilege for binding new plans, or BIND privilege for the plan if binding an existing plan again
- EXECUTE privilege for the read-only IMS DPROP collection ID
- EXECUTE privilege for the propagating collections or for the packages of individual SQL update modules and Propagation exit routines

If you are not using DB2 package bind, the plan owner must have the following privileges to bind the plans of propagating applications:

- BINDADD privilege for binding new plans, or BIND privilege for the plan if binding an existing plan again
- SELECT privilege for the IMS DPROP directory tables
- SELECT, UPDATE, INSERT, and DELETE privileges for the propagated tables

  After binding the plans of the propagating programs and receiver program, you need to grant the EXECUTE privilege for them. See "Running propagating applications" on page 125.

Chapter 8, "Binding and administering plans," on page 127 has additional information about binding DB2 plans.

Running propagating applications

Users of propagating application programs or receiver programs must have the EXECUTE privilege for these plans. You should review DB2 Administration Guide to determine the method of implementing DB2 security that best suits your needs. The techniques described in the following sub-sections are suggestions for:

- Message processing and Fast Path regions
- IMS batch and batch message processing programs
- DB2 sign-on authorization exits

Message processing and Fast Path regions

Authorization for IMS MPPs, message-driven BMPs, and Fast Path regions is usually done using IMS transaction code security. Authorization for related DB2 plans can be granted to PUBLIC. Therefore, authorization to execute the transaction can be viewed as authorization to execute the plan. This method also reduces overhead required by DB2 authorization processing.

If you are granting the EXECUTE privileges of plans of MPPs/IFPs to PUBLIC, consider preventing misuse of these plans in environments other then message processing and Fast Path regions. Specify the ENABLE keyword in the DB2 BIND command. For example, you can use the ENABLE keyword to limit the execution of a plan to the message regions of a specific IMS online system. For example:

```
BIND PLAN (planname) ... ENABLE (IMSMPP) IMSMPP (imsid)
```

You can also grant authorization to functional identifiers. Functional identifiers identify functional groups, such as an accounts payable department, to the DB2 system. If you use functional identifiers, you need a DB2 Sign-on Authorization exit routine. The exit routine associates user identifiers that need to execute propagating programs with functional identifiers.

IMS batch and batch message processing programs

You can grant authorization for batch and BMP programs to specific functional identifiers or to specific user IDs. If functional identifiers are used, you need a DB2 Sign-on Authorization exit routine.

DB2 Sign-on Authorization exits

You can use a DB2 Sign-on Authorization exit routine to associate user identifiers with functional identifiers. If you are using this exit routine, you can grant the EXECUTE privilege for the DB2 plans of propagating applications to the functional identifiers. This minimizes the number of identifiers to which authorization to execute the DB2 plan must be granted; it also reduces the administrative efforts required to maintain authorizations.

Chapter 8. Binding and administering plans

This chapter describes:

- Binding DB2 plans of:
  - The Receiver
  - The user-written receiver program
- Administering DB2 plans with and without a Resource Translation table (RTT)

You must bind DB2 plans for programs that call IMS DPROP because IMS DPROP makes SQL updates and the SQL reads the IMS DPROP directory, which is composed of DB2 tables. Bind plans for Receivers, propagating application programs, and receiver programs. You must also be authorized to use the plans.

You can bind plans with or without use of the DB2 package bind function.

Binding plans with bind package

With DB2 Version 3 Release 1 and following releases, you can use the DB2 package bind facility for the DB2 plans of your propagating applications and asynchronous receiver program.

Using the DB2 package bind facility, you can bind an individual DBRM as a *package* into a *package collection*, identified by a *collection ID*. Then, when binding the DB2 plan, you specify which packages, collection IDs, and DBRMs will be included in the DB2 plan.

Using the DB2 package bind facility has the following advantages:

- You do not need to perform another bind for the DB2 plans of propagating applications when propagation requests affecting those applications are changed or added. Instead, you only need to bind the DBRM of the affected SQL update module into a package. You reduce the number of required bind operations and simplify administration of DB2 plans.

  For example, using bind package you do not need to do both an initial and a subsequent bind for the plans of propagating applications.
- The bind for individual plans is simplified, because you do not need to specify a complete list of DBRMs that are part of the DB2 plan. You do not need to keep track of which DBRM is required in which plan. Instead, when binding a plan, you can specify the collection IDs.
- You can benefit from having different ISOLATION attributes for different packages. Packages of IMS DPROP modules should be bound with the ISOLATION level *cursor stability* (CS) to reduce the chance of DB2 enqueue conflicts on the small IMS DPROP directory tables.

  Bind packages allow you to provide a *different* qualifier for the unqualified table names of each package. You can avoid the cumbersome requirement of defining ALIASs and SYNONYMs.

The following sections present:

- Use of different collection IDs
- Job stream for binding DB2 packages
- Job stream for binding DB2 plans with bind package

Using different collection IDs

Your installation will usually use different package collections, each containing packages belonging to a specific component. For example, your installation usually has:

- One or several read-only IMS DPROP collections containing packages of IMS DPROP modules reading the IMS DPROP directory tables. Each IMS DPROP system has its own read-only IMS DPROP collection. These collection IDs are identified during IMS DPROP installation and customization.
- One or several propagating collections for SQL update modules and for user-written exit modules updating your propagated tables, for example, a collection ID for the test environment and another collection ID for the production environment.

  You need several propagating collections if you define propagation requests with unqualified table names and use the same propagation request to propagate to different tables with different qualifiers. Therefore, you create several bind packages, with different qualifiers of the same DBRM, into different collections.

Determine which package collection is used for each purpose and determine the collection IDs used to bind packages and plans. Also grant the following DB2 privileges for package collections:

- CREATE IN COLLECTION privilege for each collection. This privilege is needed to bind a package into a specific collection.
- EXECUTE privilege for entire collections. This privilege is needed to bind the plan of propagating applications if you are specifying entire collections on the PKLIST keyword of the BIND PLAN command, as recommended by IMS DPROP.
- Depending on the standards of your installation, possibly the BIND and COPY privileges for entire collections.

Job stream for binding DB2 packages

If you are using the DB2 package BIND option, you will usually bind packages required to run your propagating applications and the asynchronous receiver program, such as:

- Packages of IMS DPROP modules accessing IMS DPROP directory tables in read-only mode. These packages are usually bound into the IMS DPROP read-only collection during IMS DPROP installation.
- Packages of SQL update modules used with propagation requests belonging to generalized mapping cases and used for IMS-to-DB2 propagation. These packages are usually bound when propagation requests are created as part of MVG processing.

  If you create propagation requests with unqualified table names and use the same propagation request to propagate to multiple, identically structured tables with different qualifiers, then you can use a BIND COPY command to bind additional packages with different qualifiers.
- Packages of your Propagation exit routines. These packages are bound after creation of the tables and precompilation and compilation of Propagation exit routines.

Figure 29 on page 129 is a sample job stream for binding a package. The numbers in the figure correspond to the notes following the figure.

```
//jobname JOB
//BIND    EXEC    PGM=IKJEFT01
//SYSTSPRT  DD    SYSOUT=*
//SYSPRINT  DD    SYSOUT=*
//SYSUDUMP  DD    SYSOUT=*
//SYSTSIN   DD    *
 DSN SYSTEM(DSN)
 BIND PACKAGE(collection ID)      1                        -
      MEMBER(member)              2                        -
      LIBRARY('dbrmlib')          3                        -
      ISOLATION(xx)               4                        -
      RELEASE(COMMIT)             5                        -
      VALIDATE(BIND)              6                        -
      ACTION(REPLACE)                                      -
      QUALIFIER(qualifier)        7                        -
      OWNER(owner)                8
 /*
```

Figure 29. BIND package job stream for IMS DPROP

Notes:

1. This is the collection ID where the package is to be bound. The owner of the package must be granted the CREATE IN privilege for this collection ID.
2. This is the name of the DBRM to be bound in a package.
3. This is the library containing the DBRM used as input to the bind package.
4. Specify the ISOLATION parameter as CS (cursor stability) or RR (repeatable read) depending on the needs of the module.

   IMS DPROP packages located in the read-only IMS DPROP collection should be bound with CS.
5. Specify the RELEASE parameter as COMMIT so that DB2 resources are released at commit time. The resources can then be used for concurrent processing.
6. Specify the VALIDATE parameter as BIND so that DB2 resources used by the package are validated when the package is bound, rather than when the application runs. This improves performance of your propagating programs, especially propagating MPPs and IFPs.
7. If you specify a QUALIFIER, its value is the qualifier for any unqualified names in static SQL statements that are present in the DBRM being bound. For example, use the QUALIFIER keyword when binding the package of an SQL update module of a propagation request created with an unqualified table name.
8. The OWNER keyword specifies the owner of the package being bound. If the OWNER keyword is not present, it defaults to the primary AUTHID of the bind process.

   If the QUALIFIER keyword is not specified, then the owner is used as qualifier for any unqualified table name in static SQL statements of the DBRM being bound.

Job stream for binding DB2 plans with bind package

Figure 30 on page 130 is a sample job stream for binding a plan of a Receiver, a propagating application, or receiver program. The numbers in the figure correspond to the notes following the figure.

```
//jobname JOB
//BIND    EXEC    PGM=IKJEFT01
//SYSTSPRT  DD    SYSOUT=*
//SYSPRINT  DD    SYSOUT=*
//SYSUDUMP  DD    SYSOUT=*
//SYSTSIN   DD    *
 DSN SYSTEM(DSN)
 BIND PLAN (planname)                                  -
      ACTION(REPLACE)                                  -
      PKLIST(propag-colid2.*,        1                 -
             appl-colid2.*,          2                 -
             ...                                       -
             dprop-colid3.*)         3                 -
      LIBRARY('applpgm.dbrmlib1',    4                 -
              'applpgm.dbrmlib2')                      -
      MEMBER(appl_dbrm6              4                 -
             appl_dbrm7)                               -
      ISOLATION(CS)                  5                 -
      RELEASE(COMMIT)                5                 -
      ACQUIRE(USE)                   5                 -
      VALIDATE(BIND)                 5                 -
      QUALIFIER(qualifier)           5                 -
      OWNER(owner)                   6                 -
      ENABLE (IMSMPP) IMSMPP (imsid) 7                 -
      RETAIN
/*
```

Figure 30. BIND plan job stream when using packages

Notes:

1. *propag-colid2* identifies a propagating collection. In this example, there is only one propagating collection. It contains the packages of the SQL update modules of propagation requests for the Receiver, application, or receiver program. It also contains the packages of any user-written IMS DPROP exit routines that issue SQL calls.

   Depending on how your system is organized, you might need to provide collection IDs of more than one propagating collection in the PKLIST keyword.

   In this example, PKLIST identifies entire collection IDs rather than individual packages, as specified by coding a *.** after the collection ID. Specifying the entire collection is convenient when you do not need to know which propagation request is used by each plan and application. However, specifying *.** requires that the owner of the DB2 plan have the EXECUTE privilege on the entire collection.
2. *appl-colid2* is the name of a collection ID containing the packages of user-written application modules that issue SQL statements.
3. *dprop-colid3* identifies the IMS DPROP read-only collection. This collection contains the packages of IMS DPROP modules reading the IMS DPROP directory tables. Each IMS DPROP system has its own read-only collection. If necessary, ask your system administrator which system and read-only collection applies to your plan.

   The sequence in which the collection IDs are specified can affect performance. Specify the collection IDs of the most frequently run packages first in the PKLIST keyword. Because the packages of the read-only IMS DPROP collection are run infrequently, the collection ID is the last collection in PKLIST.
4. In this example, some application DBRMs are also bound directly into the plan. Therefore, the LIBRARY keyword identifies the names of libraries containing the DBRMs. And the MEMBER keyword identifies the name of the DBRMs.

5. The keywords ISOLATION, RELEASE, ACQUIRE, VALIDATE, and QUALIFIER only affect the DBRMs that are included directly into the plan, not DBRMs that have been bound into packages. Options for the packages are specified at bind package time.
6. The OWNER keyword specifies the owner of the plan being bound. If the OWNER keyword is not present, it defaults to the primary AUTHID of the binder.
7. In this example, the ENABLE keyword restricts use of the plan to IMS MPP programs of a specific IMS online system. The example assumes the plan of an MPP is bound and assumes that the installation uses IMS transaction security and grants the EXECUTE privilege of the plan to PUBLIC to improve performance. Restricting use of the plans to MPPs prevents accidental and intentional misuse of the plan in environments that are not protected by IMS transaction security.

   When binding the plan of BMPs, batch programs, and the receiver program, you either provide different ENABLE specifications or omit the ENABLE specifications.

Binding plans without bind package

This section describes binding the plans of Receivers, application programs, and asynchronous receiver programs without using the DB2 package BIND option. Topic in this section are:

- Binding the Receiver
- Binding the user asynchronous receiver program
- Job stream for binding DB2 plans without bind package
- DB2 ALIAS and SYNONYM statements

Binding the Receiver

The DB2 plan for the Receiver consists of the DBRMs of IMS DPROP modules as well as the original DBRMs of the Receiver. If you are doing LOG-ASYNC propagation, the Receiver that calls RUP must be bound with:

- DBRMs for SQL update modules for propagation requests
- DBRMs for IMS DPROP exit routines issuing SQL statements
- DBRMs for RUP's access to the IMS DPROP directory tables
- DBRMs for the receiver program's access to DB2 tables

Binding the user asynchronous Receiver program

The DB2 plan for a Receiver program for user asynchronous propagation consists of the DBRMs of IMS DPROP modules and the original DBRMs of the receiver program. If you are doing user asynchronous propagation, the receiver program that calls RUP needs to be bound with:

- DBRMs for SQL update modules for propagation requests
- DBRMs for IMS DPROP exit routines issuing SQL statements
- DBRMs for RUP's access to the IMS DPROP directory tables
- DBRMs for the receiver program's access to DB2 tables, if any

Job stream for binding DB2 plans without bind package

Figure 31 on page 132 is a composite BIND job stream showing all DBRMs that might be required to bind an application or receiver program. The numbers in the figure correspond to the notes following the figure.

```
//jobname JOB
//BIND    EXEC    PGM=IKJEFT01
//SYSTSPRT  DD    SYSOUT=*
//SYSPRINT  DD    SYSOUT=*
//SYSUDUMP  DD    SYSOUT=*
//SYSTSIN   DD    *
DSN SYSTEM(DSN)
BIND PLAN (planname)                                    -
     LIBRARY('DPROP.EKYDBRM'                           -
             'applpgm.dbrmlib'        1                 -
             'sqlupdt.dbrmlib'                          -
             'exitr.dbrmlib'                            -
             'receiver.dbrmlib')                        -
     MEMBER(EKYX120X                  2                 -
            EKYGC001                  3                 -
            EKYGH001                                    -
            EKYGM001                                    -
            ...                                         -
            appl_dbrm1                                 -
            appl_dbrm2                                  -
            ...                                         -
            pr1                       4                 -
            pr2                                         -
            ...                                         -
            prnn                                        -
            exitr1                    5                 -
            exitr2                                      -
            ...                                         -
            exitrnn                                     -
            receiver_dbrm1            6                 -
            receiver_dbrm2)                             -
     ISOLATION(CS)                    7                 -
     RELEASE(COMMIT)                  8                 -
     ACQUIRE(USE)                    9                 -
     VALIDATE(BIND)                  10                 -
     ACTION(REPLACE)                                    -
     QUALIFIER(qualifier)                               -
     OWNER(owner)                                       -
     ENABLE (IMSMPP) IMSMPP (imsid)  11                 -
     RETAIN
/*
```

Figure 31. BIND plan job stream without packages

Notes:

1. This is a concatenation of the IMS DPROP-supplied DBRM library and any user DBRM libraries associated with included DBRM members.
2. The EKYX120X DBRM name must be coded as shown. EKYX120X uses unqualified table names to reference IMS DPROP directory tables. The bind process sets the qualifier of the IMS DPROP directory table names. If the qualifier set by BIND does not satisfy your requirements, you might want to use a DB2 CREATE SYNONYM or CREATE ALIAS statement for the propagated tables. See “DB2 ALIAS and SYNONYM statements” on page 133 for more information.
3. The following DBRM names might vary depending on your installation:
   - EKYGC001
   - EKYGH001
   - EKYGM001

   These DBRM names are for specific IMS DPROP systems you have defined. Each IMS DPROP system has its own copy of these modules. The first IMS

DPROP system uses the suffix 001, the second IMS DPROP system uses 002, the third IMS DPROP system uses 003, and so on. You need to know which system you are using to specify the appropriate suffix for these two DBRMs.

4. For IMS-to-DB2 propagation, there are DBRMs for SQL update modules of all propagation requests for the application or receiver program. This does not apply to the initial bind.
5. There are DBRMs for any user-written exit routines that issue SQL statements. This does not apply to the initial bind.
6. These receiver DBRMs are necessary if the receiver program issues any SQL calls.
7. Specify the ISOLATION parameter as CS (cursor stability) to reduce contention on the IMS DPROP directory tables and improve concurrent access to them.
8. Specify the RELEASE parameter as COMMIT to release DB2 resources at commit time; you can then use the resources for concurrent processing.
9. Specify the ACQUIRE parameter as USE so that DB2 locks and resources are acquired when used, instead of when allocated.
10. Specify the VALIDATE parameter as BIND so that DB2 resources used by the plan are validated when the plan is bound, instead of when the application begins.
11. In this example, the ENABLE keyword restricts use of the plan to IMS MPP programs of a specific IMS online system. The plan of an MPP is bound and the installation uses IMS transaction security and grants the EXECUTE privilege of the plan to PUBLIC to improve performance. Restricting use of the plans to MPPs prevents accidental and intentional misuse of the plan in environments that are not protected by IMS transaction security.

    When binding the plan of BMPs, batch programs, and the receiver program, you do not specify ENABLE.

DB2 ALIAS and SYNONYM statements

You usually use DB2 aliases and synonyms in installations where the DB2 package bind is not used. The installations include DBRMs directly into their DB2 plans.

The following static SQL statements issued by IMS DPROP have unqualified table names:

- Most SQL statements issued to access IMS DPROP directory table DPRMASTER
- Propagating SQL statements issued by SQL update modules if you specify during propagation request definition unqualified table names for the propagated tables

During the bind process, the qualifier for these SQL statements is set based on either the:

- Authorization ID used for the bind process
- Authorization ID on the QUALIFIER keyword
- Optional OWNER keyword of the BIND command in DB2

Sometimes the qualifier set by BIND is not convenient. For example, the qualifier set by BIND might be different from the qualifier for your IMS DPROP directory tables or the propagated tables. Before the bind you can use a DB2 CREATE ALIAS or CREATE SYNONYM statement to match the bind and IMS DPROP qualifier. Figure 32 on page 134 shows the two-step BIND process when you create

aliases or synonyms before bind.

```
             Qualifier                      Translation
             SANDY                          by ALIAS or
             set by                         SYNONYM
             BIND                           mechanism
DPRMASTER --------------> SANDY.DPRMASTER -------------->PROD.DPRMASTER
```

Figure 32. Two-Step BIND Process

First, BIND sets the qualifier for the unqualified SQL statements based on either the:
- Value of the QUALIFIER keyword
- Value of the optional OWNER keyword
- Authorization ID used for the bind

In this example, the qualifier is SANDY.

Then use the ALIAS or SYNONYM mechanism to translate SANDY.DPRMASTER into the table name specified when you issued the CREATE ALIAS or CREATE SYNONYM statement: PROD.DPRMASTER.

Using the CREATE ALIAS statement

Before the bind process, you can use the DB2 CREATE ALIAS statement to create two-part alias names for the DPRMASTER directory table. For the first part of the alias, use the qualifier that is set by the bind process. For the last part of the alias, use the unqualified name of the IMS DPROP directory table, DPRMASTER. See Figure 33.

```
CREATE ALIAS SANDY.DPRMASTER FOR PROD.DPRMASTER
```

Figure 33. Using the DB2 CREATE ALIAS statement

In the example:
- The qualifier of the IMS DPROP directory tables is PROD
- The qualifier set by a bind process will be SANDY

Therefore, the CREATE ALIAS statement creates the alias SANDY.DPRMASTER for the PROD.DPRMASTER table.

When the bind of the plan of a propagating application sets SANDY as the qualifier for unqualified SQL statements, access to DPRMASTER is qualified as SANDY.DPRMASTER. If the alias is created before the bind, ALIAS processing translates SANDY.DPRMASTER into PROD.DPRMASTER. Therefore, unqualified IMS DPROP SQL statements for the DPRMASTER table access the PROD.DPRMASTER table.

If you define propagation requests with unqualified table names, you might also want to create aliases for the propagated tables.

Using the CREATE SYNONYM statement

Before the bind process, you can use the DB2 CREATE SYNONYM statement to create a synonym for the DPRMASTER directory table. Use the unqualified name of the IMS DPROP directory table, DPRMASTER, as the synonym. The authorization

ID used to issue the CREATE SYNONYM statement should be the same as the qualifier later set by the bind process.

```
CREATE SYNONYM DPRMASTER FOR PROD.DPRMASTER
```

Figure 34. Using the DB2 CREATE SYNONYM statement

In the example:
- The qualifier of the IMS DPROP directory tables is PROD
- The qualifier set by an eventual bind process will be SANDY

Therefore, the above CREATE SYNONYM statement should be issued by the authorization ID SANDY.

When the bind of the plan of a propagating application sets SANDY as the qualifier for unqualified SQL statements, access to DPRMASTER is qualified as SANDY.DPRMASTER. If you issue the CREATE SYNONYM statement before the bind by the authorization ID SANDY, then during the bind SYNONYM processing translates the qualified name SANDY.DPRMASTER into PROD.DPRMASTER. Therefore, unqualified IMS DPROP SQL statements for the DPRMASTER table access the PROD.DPRMASTER table.

If you define propagation requests with unqualified table names, you might also want to create synonyms for the propagated tables.

Administering DB2 plans with or without a Resource Translation Table (RTT)

You can administer plans for online or batch regions in two ways:
- Use of an RTT so that one DB2 plan can be shared by multiple propagating application programs. The RTT associates application programs and plan names.
- Use of a different DB2 plan for each application program.

Use the method you currently have in place. If you are new to IMS/DB2 mixed-mode applications, determine which procedure works best at your installation.

An advantage of using RTTs is you must define fewer DB2 plans in the system. However, when you create new applications, you must update the RTT source to associate new programs with the name of the plan. You use the DB2 DSNMAPN macro to generate RTT entries. After the source is altered, you must recompile and link-edit the RTT.

For IMS batch regions, you can define the DB2 connection in the //DDITV02 file or in a subsystem member (SSM).

For more information on RTTs and how to construct and maintain them, refer to *DB2 Administration Guide*.

Chapter 9. Extracting and loading data for IMS-to-DB2 propagation

This chapter explains the process of extracting data from an IMS database and loading it into a target DB2 table using DataRefresher or your own program. Topics include:

- An overview of the extract and load process
- Suggestions for preventing updates to IMS databases
- A description of doing the extract and load with DataRefresher
- A description of doing the extract and load with your programs
- Considerations when IMS and DB2 reside on different MVS images
- LOG-ASYNC extract and load considerations

For details on how to code an extract request for DataRefresher refer to the *IMS DataPropagator for z/OS Reference*. For details on how to code an extract request using your own program see *Customization*.

Overview of the extract and load process

You usually perform extract and load after creating propagation requests. To extract data from IMS and load it into target DB2 tables:

- First, prevent updates to IMS databases using the SCU or the appropriate IMS commands.
- Next, extract and load data into DB2 tables using DataRefresher DEM or a user extract program. Or you can load DB2 rows by running your IMS database load programs in PROP LOAD mode; although this method is not efficient.
- Then, run DB2 utilities such as COPY and RUNSTATS to establish a common point of recovery for IMS and DB2. You might also want to make image copies of the IMS databases.

For large databases, a substantial amount of time might be required to:

- Do an image copy of the IMS databases
- Extract data from an IMS database
- Load the data into DB2 tables
- Build index entries (part of the DB2 load process)
- Do an image copy of the loaded tables
- Execute the RUNSTATS utility against the DB2 tables

You should plan for the IMS databases being propagated to not be available during the extract and load phase.

Preventing updates to IMS databases

If updates are made during the extract and load, data inconsistencies can occur. If you have registered your databases in DBRC, use SCU to prevent updates. The SCU works through the DBRC to prevent or permit updates. If you have not registered your databases in DBRC, you must use alternative methods to prevent the databases from being updated during the extract and load phase. Using the SCU and alternatives to using SCU are described in this section.

Using Status Change Utility (SCU)

You can use the SCU to make the source IMS database available in read-only mode. Read-only mode prevents data from being updated during the extract. Use the SCU READON control statement to set the database status to read-only.

After data has been extracted and loaded into DB2, call the SCU again to activate propagation and make the database available for updates. You can use the READOFF control statement, which turns off or resets the read-only status so that the database is available for updates.

Now, any changes made to IMS data to be propagated are propagated to DB2.

The extract and load phase is considered complete only after the DB2 COPY and RUNSTATS utilities have been run against the loaded table. Therefore, you should call SCU with ACTIVATE and READOFF control statements only after running DB2 COPY and RUNSTATS.

Using the SCU to control access to IMS databases requires that:

- IMS databases are full function, not DEDBs
- Databases are registered in DBRC
- DBRC share control are used

If any requirement is not met, the SCU issues warning messages but takes no other action.

Related Reading: For more information on how to register databases in DBRC and use share control, refer to *IMS/ESA Utilities Reference: Database Manager*.

Alternative to using SCU

If you do not use DBRC for controlling access to your IMS databases, you cannot use the SCU to prevent updates during the extract and load phase. Instead, you must use other methods to prevent such updates.

For full-function databases in an IMS online environment, you can:

- Use the IMS /DBD (or /DBDUMP) command to prevent transactions or programs running under the IMS control region from updating the database. We recommend that you force the end of volume of the IMS log so that a recovery point is established for the databases. You can force the end by omitting the NOFEOV parameter from the /DBD command.
- Perform the extract and load process after update activity has quiesced. Copying the DB2 tables and executing RUNSTATS against the tables is part of the extract and load phase.
- Restart the databases using a /STA DB command: ACCESS=UP, for update, or ACCESS=EX, for exclusive use.

These methods do not protect against concurrent batch updating jobs or other concurrent online systems.

Related Reading: Refer to *IMS/ESA Operations Guide* for specific information on the /DBD and /STA commands.

Performing extract and load with DataRefresher

Extracting and loading propagated data is simplified if you use DataRefresher. With DataRefresher, mapping and conversions are identical to those done by IMS DPROP during propagation.

When you use DataRefresher in the extract and load phase of propagation, the following events occur:

- When extracting with the DataRefresher DEM, IMS-to-DB2 mapping is based on information stored in the DataRefresher EXTLIB and FDTLIB.
- DEM calls the IMS DPROP Map Capture exit (EKYMCE00). When called to extract a propagated, DBRC-registered, full-function database, EKYMCE00 verifies that the database is in read-only status and cannot be concurrently updated by any IMS subsystem. Verification is only possible if DBRC share control is in effect. To perform the validation, EKYMCE00 internally invokes the IMS DBRC utility.
- The DEM provides the extracted and mapped IMS data to the DB2 LOAD utility.

  If Segment or Field exit routines is specified, the DEM calls them during the extract process. The DEM calls are an important part of achieving mapping and conversion identical to those used during propagation.

  The DEM does not call Propagation exit routines during the extract process. The DEM, not your Propagation exit routines, performs mapping and conversion during extract.

Related Reading: Refer to the *IMS DataPropagator for z/OS Reference* for detailed information on how to code an extract request for DataRefresher.

The extract and load process is illustrated in Figure 35 on page 140.

Extract and Load Process Using DataRefresher

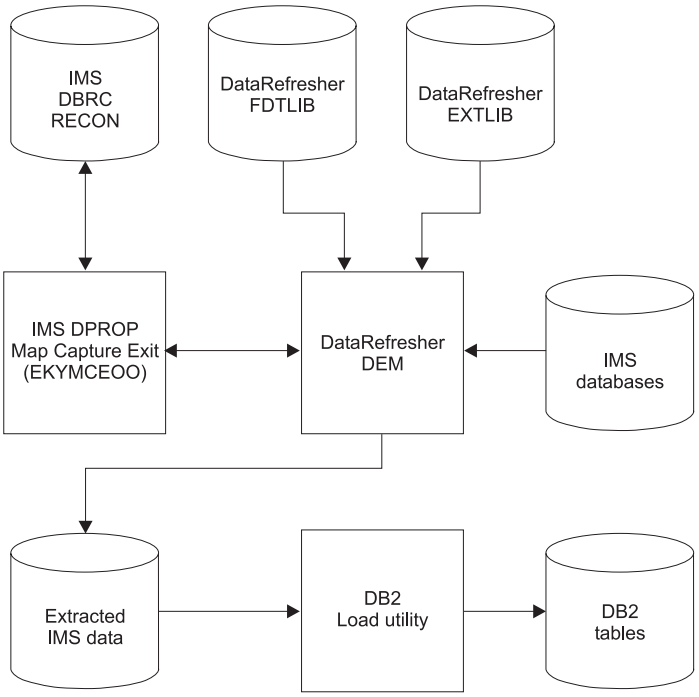


Figure 35. Extract and load process using DataRefresher. After the tables have been loaded, you should run the RUNSTATS utility and copy the tables.

Consider the following information when you use the DataRefresher DEM to extract propagated data:

- IMS DPROP functions are used during the extract process. You must modify the DEM JCL to include the data sets and libraries required by IMS DPROP. Refer to the *IMS DataPropagator for z/OS Reference* for more information on this subject.
- You usually request that the DataRefresher DEM create the control statements for the DB2 LOAD utility by providing a CD keyword on the DataRefresher SUBMIT command. Requesting that the DEM create the load control deck reduces effort and also eliminates one source of potential errors.
- You use the DataRefresher SUBMIT command to request that the DataRefresher DEM create a job to execute the DB2 LOAD utility. Specify the ddname of a file containing skeleton JCL for the DB2 LOAD utility on the JCS keyword of the SUBMIT command.
- For improved performance, you can process DataRefresher extracts of all segments of the same IMS database with a single pass through the database. This practice is called batching DataRefresher extract requests. Batching can save a considerable amount of time and processing. To benefit from extract request batching, you must provide multiple //DXTOUTn DD JCL in the DataRefresher DEM job stream. You must also ensure that all batched extract requests and propagation requests be based on DXTVIEWs that use the same DXTPCB.

  Batching extract requests that belong to generalized mapping cases and extract requests that belong to user mapping cases have restrictions. Refer to the *IMS DataPropagator for z/OS Reference* for a description of the restrictions.

- If you are extracting and loading data into multiple tables, you might want to run the DB2 LOAD utility jobs in parallel to reduce the amount of elapsed time required to load the tables.

  If the DB2 tables are involved in RIRs, you can:

  – Specify ENFORCE NO on the USERDECK keyword of the DataRefresher SUBMIT command. With ENFORCE NO, the DB2 LOAD utility does not check referential integrity constraints during load processing. When the target table spaces are loaded, DB2 places them in a check-pending state.
  – Run the DB2 CHECK utility after completing all DB2 load jobs.

Performing extract and load with your programs

If you do not use DataRefresher to extract data from the IMS database you want to propagate, you must provide programs that extract the IMS data and load the DB2 tables. You can use one of the following methods:

- **Method 1:** Write a program that extracts and maps the IMS data and creates an input file for the DB2 Load utility. Then run the DB2 Load utility to load the data into your DB2 tables. For information on the DB2 Load utility and its input file, refer to the *DB2 Command Reference* and *DB2 Utility Guide and Reference*.

  Use this method when you are loading a lot of data into DB2. You have better performance because loading a lot of DB2 data is usually more efficient with the DB2 Load utility than with SQL insert statements.
- **Method 2:** Write a program that extracts and maps IMS data and issues SQL insert statements to insert the data into DB2 tables.

  Using SQL insert statements is usually slower than using the DB2 Load utility. However this method works well for small DB2 tables.
- **Method 3:** Execute your IMS database load programs in PROP LOAD mode. Provide a PROP LOAD control statement in the //EKYIN file allocated to the job step doing the IMS database load. RUP then maps and propagates the IMS inserts to the DB2 tables. With this method, the IMS-to-DB2 mapping, conversion, and propagation is done by RUP.

  RUP uses SQL insert statements to store the data into the DB2 tables. For extension segments of mapping case 2, RUP uses SQL update statements. Using SQL insert statements is usually slower than using the DB2 Load utility. However, this method works well for small DB2 tables.

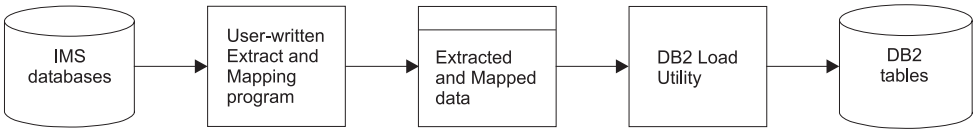
If you use methods 1 and 2, your programs must provide the IMS-to-DB2 mapping and conversion logic. The mapping and conversion must be compatible with IMS DPROP's mapping and conversion. If you want to use the CCU to verify data consistency, your mapping and conversion must be identical to IMS DPROP's.

Figure 36 on page 142 is an overview of the three methods of doing the extract and load using your programs.

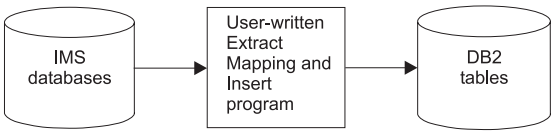
Related Reading: See the *IMS DataPropagator for z/OS Reference* for information on how to code an extract request using MVG input tables without DataRefresher.

Extract and Load Process with User-Written Programs

Method 1: User-provided Extract/Mapping program and the DB2 Load utility



Method 2: User-provided Extract/Mapping/SQL Insert program



Method 3: User-provided IMS DB Load program

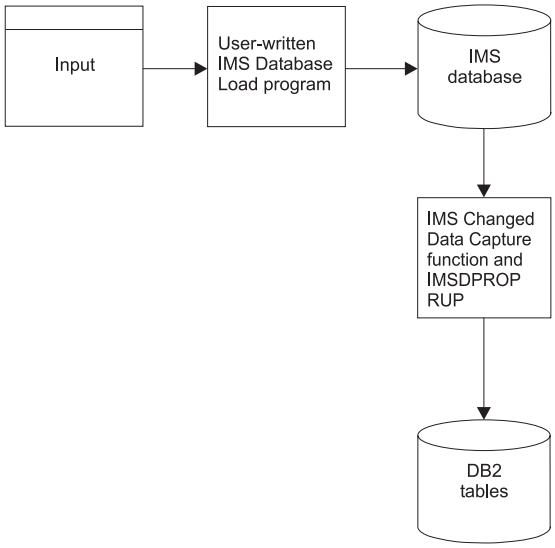


Figure 36. Extract and load process with user-written programs

When IMS and DB2 reside on different MVS images

You can use one set of DataRefresher definitions to both create propagation requests and extract and load data only when IMS and DB2 are on the same MVS image. If IMS and DB2 are on different MVS images, then DataRefresher must be available on both MVS images. The mapping definitions and any exit routines you define must also be available on both MVS images. You can:

- **Create the propagation request**. Run the DataRefresher UIM with the IMS DPROP MCE routine EKYMCE00 on the MVS image where DB2 resides.

  The IMS DPROP:

  – Directory tables must be located on the DB2 system
  – MCE requires access to the IMS DBDLIB. If the IMS DBDLIB is not shared between the IMS and DB2 system, you should copy the DBDLIB to the DB2 system.
- **Extract the IMS data**. You first need to create a set of mapping definitions on the IMS system. To do so, run the DataRefresher UIM (without definition of the IMS DPROP MCE) on the IMS system. Then extract the IMS data by running the DataRefresher DEM on the IMS system.

  You can use the data extracted by the DEM to load the DB2 tables on the other system.

LOG-ASYNC extract and load considerations

You must prepare IMS and DB2 databases for the load and extract procedure.

DB2 database load using IMS data

If you are maintaining duplicate database copies in IMS and DB2, you must initially populate the DB2 copy prior to propagation, using DataRefresher or another extract program. The IMS database must be quiesced in order to get an extract with integrity. You can set IMS full function databases to read-only, using the IMS DPROP SCU. Take DEDBs offline for the extract process.

While the IMS database is quiesced, the SCU timestamp marker facility creates a quiesce timestamp for the database. The timestamp is assigned to the propagation groups containing the database and is used as a valid start time for the selection of log records for the database.

The start timestamp is useful, for example, when the IMS DBDGEN is performed. When the following conditions are true, you do not want to select the ACDC log data, nor data starting from some point after the database is made available for update post extract:

- The database is set to READONLY
- IMS DLIBATCH jobs update the IMS database
- ACDC captures data and creates ACDC log records

You want the start timestamp for the selection of ACDC log data to be during DBDGEN, when the IMS and DB2 data are synchronized.

IMS database load considerations

IMS does not provide an option with the DBD EXIT parameter to allow you to capture or not capture IMS load data. You can, however, perform the following actions if you do not want to propagate data from a load operation:

- Alter IMS DBDs for ACDC after load operations are complete to activate the EXIT keyword.
- When performing load operations at some point after changing the IMS DBDs for ACDC, quiesce the database after the load operation is complete, synchronize with DB2 tables, create a quiesce TSM, and assign this quiesce time as the database start time.

Part 4. Propagating data with IMS DPROP

Chapter 10. Performing LOG-ASYNC propagation

This chapter describes how to use the main components of LOG-ASYNC propagation, which are:

- A Selector, the PRDS Registration utility (PRU), and a Receiver, run regularly:
  - The Selector retrieves propagation log records from IMS logs and writes them to sequential files called propagation request data sets (PRDSs)
  - The PRDS Registration utility registers each PRDS so that the Receiver can process them
  - The Receiver retrieves the log records from the PRDSs and updates the target DB2 tables
- Other components that are run as required to set up the environment for the Selector and the Receiver

Concept of propagation groups and Receivers

This section describes:

- Propagation groups
- Receivers
- Propagation group control data
- PRDS sequencing

Propagation groups

IMS DPROP uses propagation requests to map the IMS data to DB2. For generalized mapping, each propagation request should propagate to only one DB2 table. If more than one propagation request is to propagate to the same table, each of the propagation requests should be bound into a different plan that specifies a unique table identifier. All IMS DPROP propagation requests are stored in the propagation request table (DPRPR) in the IMS DPROP directory.

IMS DPROP allows you to collect a set of propagation requests into a propagation group. A propagation group is a subset of the propagation requests in the propagation request table. You can define as many propagation groups as you like, but an individual propagation request can be associated with only one propagation group.

In general, you define propagation groups when a DB2 application or set of DB2 applications requires data from particular IMS segments. Updates to the data are propagated to the DB2 tables through a propagation group.

From the DB2 application's point of view, *all* updates to the IMS data associated with a propagation group must be propagated to DB2. Therefore, when the Selector runs, it collects all the committed IMS log records for all propagation groups that are to be processed on the current execution and writes them to the appropriate PRDSs.

If the Selector cannot collect all the committed IMS log records for all propagation groups that are to be processed, it issues an error message and stops.

Receivers

You do not directly define a set of propagation requests to a propagation group. Instead, you use the Receiver. The Receiver allows parallel processing of multiple

propagation requests. You assign one or more propagation requests to each Receiver. And one or more Receivers process a propagation group. A Receiver propagates the IMS updates for a set of propagation requests in a propagation group, as shown in Figure 37.

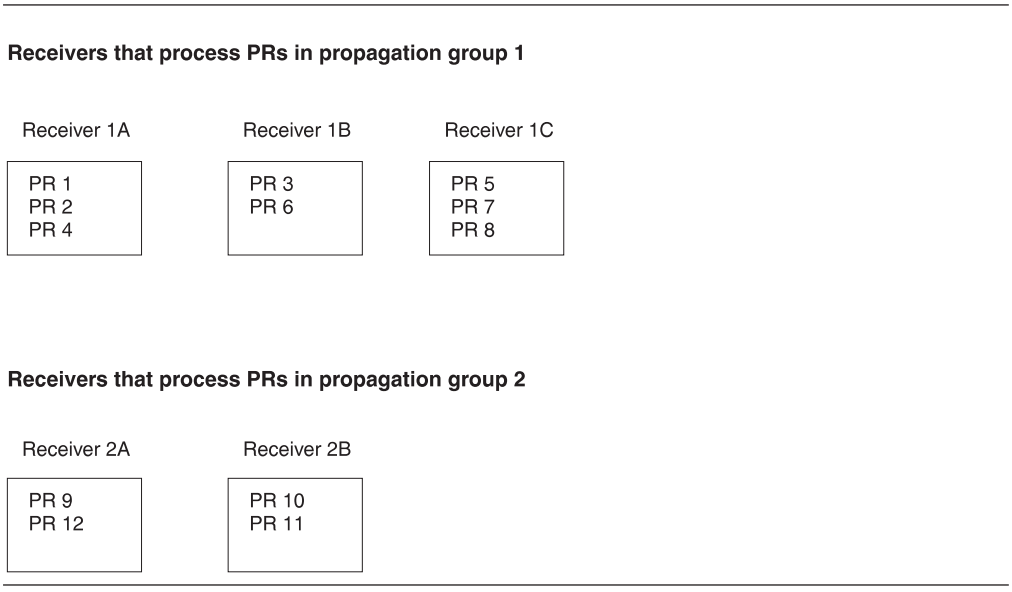


Figure 37. Relationship between propagation groups, Receivers, and propagation requests

In Figure 37:

- Each of the 12 propagation requests in the IMS DPROP directory propagation request table (DPRPR) is assigned to only one Receiver
- Each of the five Receivers in the IMS DPROP directory RCT (DPRRCT) processes only one propagation group
- Each of the two propagation groups is processed by more than one Receiver

Refer to the *Reference* for details of the IMS DPROP directory tables.

A propagation group can be processed by many Receivers. *Each propagation request in the propagation group is assigned to only one Receiver*.

You might want to use more than one Receiver to process a propagation group either to:

- Speed up the propagation of IMS updates to DB2, taking into account existing DB2 target table Referential Integrity Constraints (RIRs)
- Reassign a failed propagation request to another Receiver

If a Receiver encounters an error while processing a propagation request, you can reassign the failed propagation request to another Receiver and use:

- The first Receiver to process the remaining propagation requests
- The second Receiver to process the failed propagation request when the problem has been resolved

Propagation group control data

You can use the IMS DPROP utilities described in “Creating, modifying, and deleting propagation groups” on page 98 to:

1. Define the propagation groups, in terms of Receivers and propagation requests, at the target DB2 site. You store the control data in the following IMS DPROP directory tables:

   Receiver Control Table (RCT)
   Contains one row for each Receiver and the propagation group that it can process.

   Propagation request control table (PRCT)
   Contains one row for each propagation request assigned to each Receiver.

2. Store the propagation group definitions in the Selector control file (SCF) at the source IMS site, to allow the correct IMS logs to be selected. The SCF is a VSAM key sequenced data set (KSDS) that stores the following record types:

   0200 (database) record
   One for each database that sources data for a propagation group.

   0300 (propagation group) record
   One for each propagation group.

   0302 (propagation group/database) record
   One for each propagation group and database pair defined.

Related Reading: See the *Reference* for details of the SCF and the IMS DPROP directory.

PRDS sequencing

In IMS DPROP synchronous propagation, DB2 updates are applied in the correct order because DB2 updates are applied within the same unit of work as the corresponding updates to IMS,

However, in LOG-ASYNC propagation, IMS updates are first logged, then copied by the Selector to a PRDS, and at some later time applied to DB2. IMS DPROP helps apply DB2 updates in the same order as the corresponding IMS updates by having the Selector write the propagation log records to each PRDS in the same chronological order as they were committed in IMS. When the Receiver processes a PRDS, it applies the updates contained in it in the same order.

You can run the Selector several times for a propagation group (producing a PRDS for each execution) before you run the Receiver for the propagation group. To ensure that the Receiver processes the PRDSs in the correct order, the Selector assigns a sequence number (the next highest integer) to each PRDS it creates and the Receiver processes PRDSs in order of increasing sequence number.

The combination of propagation group ID and sequence number uniquely identifying each PRDS are stored in:
- The PRDS
- A 0305 (propagation group/stop time) record in the SCF
- The PRDS register table

Each time the Selector creates a PRDS, it writes a 0305 (propagation group/stop time) record to the SCF containing the propagation group ID, the sequence number, and a Selector complete flag set to **Y** to indicate successful processing. The Selector also writes a header record to the PRDS containing the propagation group ID and the sequence number.

You must use the PRDS Registration utility (PRU) to register each PRDS in the PRDS register table, before the PRDS can be processed by a Receiver. The PRU adds a row containing the propagation group ID, the sequence number from the PRDS header, and the PRDS data set name.

The Receiver chooses which PRDS to process by adding 1 to its own sequence number count and using the resulting value to locate the PRDS data set name in the PRDS register table

There is a link that exists between 0305 records in the SCF and rows in the PRDS register table. For correct Receiver processing of PRDSs, you must have a row in the PRDS register table for each 0305 record in the SCF whose Selector complete flag is set to **Y**.

Figure 38 on page 153 illustrates the main IMS DPROP directory tables, SCF records and relationships between the two.

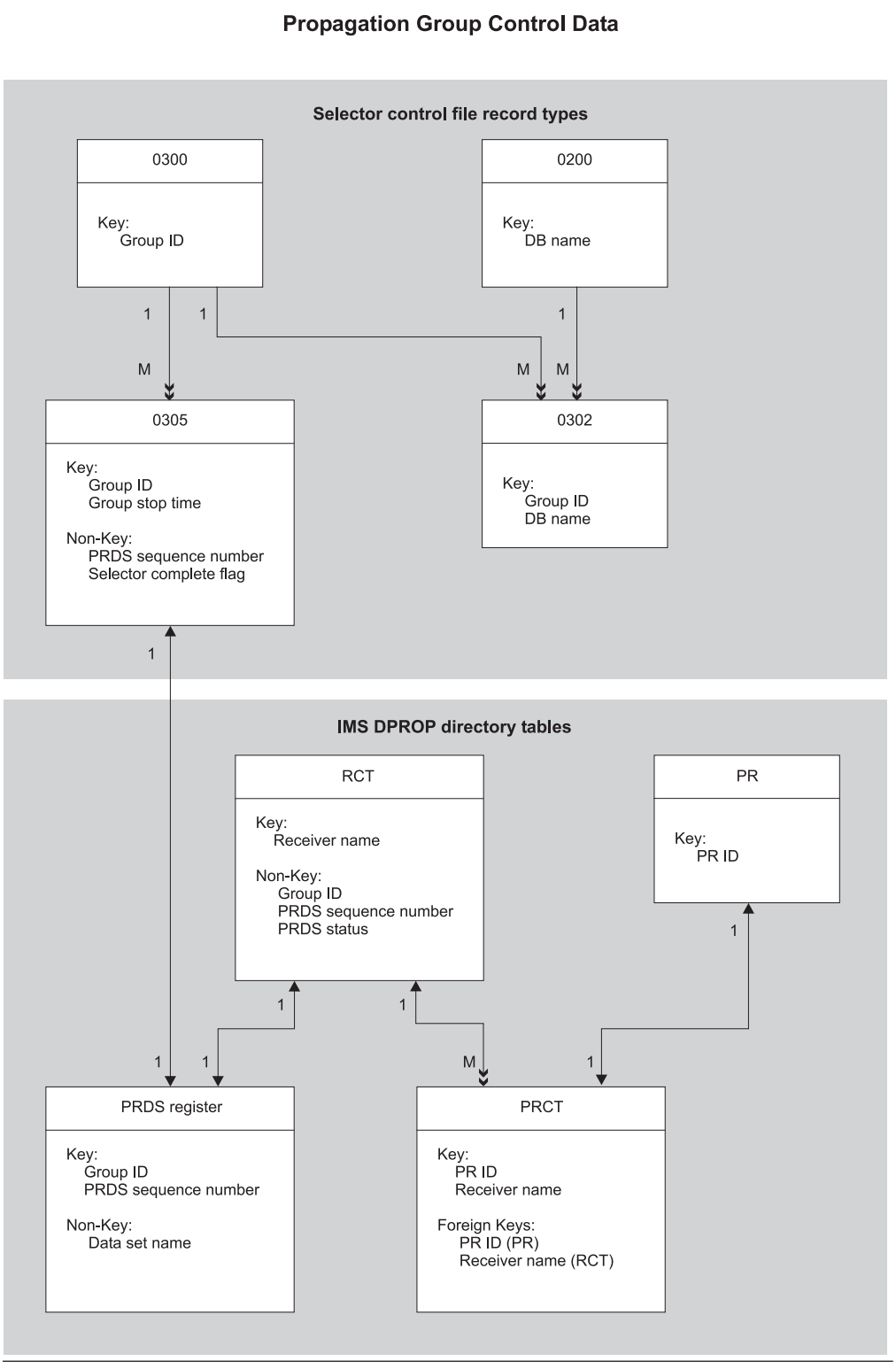


Figure 38. Propagation group control data

Using the Selector

This section describes how to use the Selector to retrieve propagation log records from the IMS logs and write them to PRDSs.

The following sections discuss the Selector in relation to:

- Input and output
- Processing
- Writing the PRDS headers
- Selecting the propagation log records
- Processing propagation group stop times
- Processing Stop Times
- User interaction
- Output messages
- Failure and recovery
- Return codes and error conditions
- Recovering from a failure

Selector input and output

Figure 39 illustrates the sources of input to the Selector and the outputs created by the Selector.

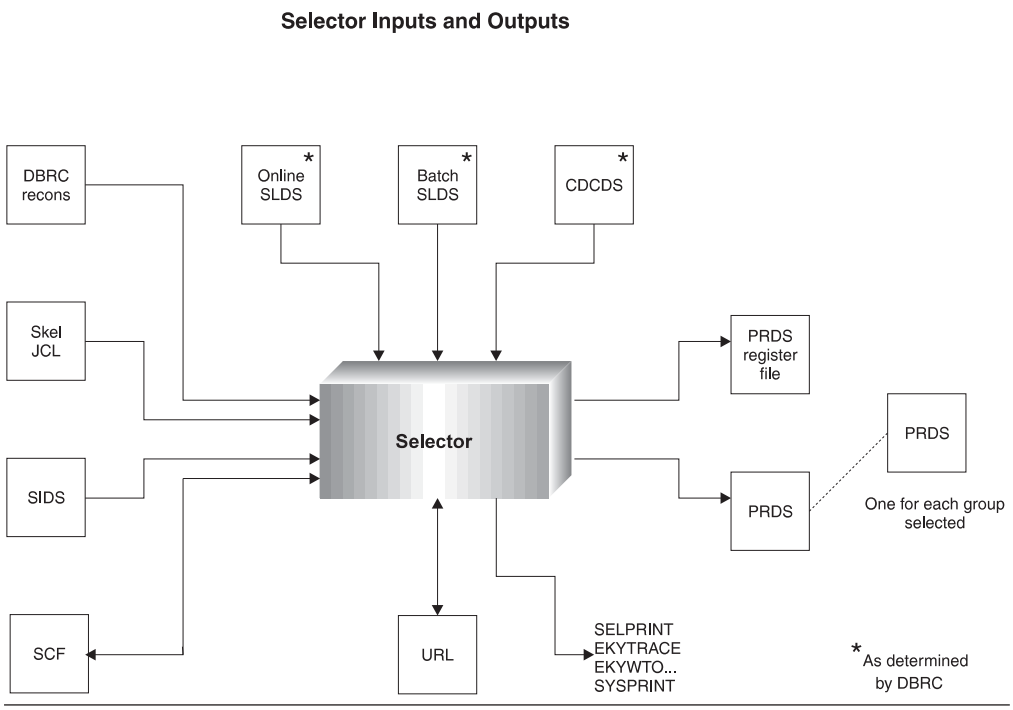


Figure 39. Selector inputs and outputs

The following list describes the input to the Selector:

Input	Description
SCF	The SCF, which is a VSAM KSDS, contains control information such as details of propagation groups, databases, segments, fields, and timestamps required to operate the Selector.

SIDS Contains Selector control statements you provide to determine the processing options used by the Selector.

ULR The ULR data set, which is a VSAM KSDS, contains uncommitted log records.

DBRC RECONS
The Database Recovery Control Interface locates the set of IMS log data sets to use as input to the Selector. Both IMS log information and Changed Data Capture Data Set information is stored on the RECON data sets and is retrieved by the Selector using the Database Recovery Control Interface.

Skel JCL
Contains the skeletal JCL members required to collect output from Database Recovery Control. The data set is allocated as part of the Selector JCL. The Selector uses the following predefined skeletal members:

- EKY System Log Data Set: Contains all SLDSs for a particular SSID
- EKYRLDS: Contains all RLDS for a given Database Recovery Control 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
PRDSs	The PRDSs are the primary output from the Selector. The Selector creates one PRDS for each propagation group in the SCF that has been specified for selection. You must specify the JCL DD statement for the PRDS in the JCL. If you specify: • SELECT ALL, you must specify a DD statement for each propagation group in the SCF • A subset of the propagation groups in the SCF, you must specify a DD statement for those groups only The DD name for each PRDS must correspond to the propagation group. Usually, the PRDS is an MVS Generation Data Group (GDG) data set.
PRDS Register File	Contains a PRDS Register control statement for each propagation group successfully processed by the Selector.
SCF Updated	Contains updated Selector control information to indicate that the propagation group has been successfully processed.
ULR data set	Contains the log records that have not been committed when the Selector finishes running; the log records are organized by propagation group. When propagation groups are deleted from the SCF, the SCF Apply utility cleans up the data set by removing any records related to the deleted propagation groups that remain in the ULR data set. Usually, the records are either committed or aborted during the next Selector run.
Trace Files	Contains information used for debugging.
Message Files	All errors and significant events are written to the SELPRINT data set.

Selector processing

This section describes how to use the Selector to collect the IMS updates, package the updates as propagation request data sets (PRDSs), and send the PRDSs to the target. The target can be one or more Receivers on a local or a remote site.

The Selector:

1. Reads and validates the SELECT control statements.
2. Determines which propagation groups to process.
3. Determines the start and stop times.
4. Gets the list of logs that satisfy the Selector start and stop times from DBRC.
5. Gets the uncommitted log records from previous Selector executions.
6. Writes the header record in the PRDS for each propagation group.
7. Selects the propagation log records that match the timestamp and propagation group information from the logs in the list. When the Selector encounters:
   - A 9904 record, it stores the record in memory if it satisfies the propagation group and timestamp criteria.
   - A commit record, it writes the appropriate UOW (the 9904 record and the commit record itself) to the PRDS
   - An abort record, it deletes the 9904 records for the aborted UOW from memory. Therefore, the aborted UOW is not written to a PRDS.
   - SETS record with a token, it stores the record.
   - ROLS record with a token that matches the token on a SETS record, it deletes all records between the SETS and ROLS.
8. When the Selector has read all logs that satisfy a propagation group start and stop times, it:
   - Writes the trailer record to the relevant PRDS
   - Writes any uncommitted log records to the uncommitted log record data set
   - Writes a message to the audit trail
   - Updates the Selector control file
9. When the Selector has read all logs that satisfy the Selector start and stop times, it:
   - Writes a PRU REGISTER statement for the PRDS to the PRDS Register file for each propagation group
   - Completes all remaining propagation groups
   - Updates the Selector execution record in the Selector control file

Reading and validating the SELECT control statements in the Selector Input Data Set

The Selector:

1. Reads the Selector Input Data Set and extracts the propagation group and timestamp information for the propagation group that you specified on the SELECT control statements. Refer to the *Reference* for the syntax and description of the SELECT control statement.
2. Validates the propagation group and timestamp information in the Selector Input Data Set. If the Selector finds any errors, it issues an error message and stops.

Determining which propagation groups to process

The Selector reads the Selector control file and locates all propagation group entries that match the entries in the Selector Input Data Set. If you specify ALL on a SELECT control statement, the Selector designates all propagation groups that have a 0300 (propagation group) record in the SCF for propagation.

If the Selector Input Data Set contains an entry for a propagation group and the Selector cannot find a matching entry in the SCF, it issues an error message and stops.

Determining the Selector start and stop times

DBRC uses the Selector start and stop times to select the list of IMS logs that it passes to the Selector. The Selector uses the propagation group start and stop times to select the propagation log records from the IMS logs.

The Selector uses the timestamp information in the SCF and the information you specify on the SELECT control statements to determine start and stop times.

Selector start time: The Selector start time for the current run is the earliest occurrence of one of the following events:

- The first start time for the propagation groups that are to be processed on the current execution.

  The start time for each propagation group is taken from the timestamp of the earliest 0302 (propagation group/database) record whose NEWSTART flag is set to **Y**.
- The first propagation group stop time from the previous Selector execution.

Therefore, the first time the Selector runs, the first propagation group start time is used as the Selector start time. On subsequent runs the first propagation group stop time from the previous Selector execution is used as the Selector start time.

Selector stop time: If you specify a stop time for one or more of the propagation groups being processed on the current run, the Selector uses the latest of these as the Selector stop time. Otherwise, the Selector determines its own stop time.

You can use the SELECT control statements to specify one of the following items as a propagation group stop time:

A timestamp
: The Selector uses the timestamp value as the propagation group stop time.

A TSM ID
: The Selector searches the SCF for an 0305 (propagation group/stop time) record with the specified ID that has not been previously used and, if it finds one, uses the associated stop timestamp as the propagation group stop time. You can create 0305 records using the SCU CREATETSM control statement.

INTERIM
: The Selector determines its own stop time.

If you do not specify a stop value for a propagation group on the SELECT control statement, but the SCF contains one or more unused timestamps for the propagation group, the latest one is used as the Selector stop time. Otherwise the Selector determines its own stop time.

Getting the list of logs

The Selector invokes the Database Recovery Control Interface to locate the IMS logs that contain the IMS log records. The required IMS logs (closed batch logs, closed SLDSs, or closed CDCDSs) are retrieved by the DBRC Interface based on information in the DBRC RECON data sets and the following information that the Selector passes from the SCF:

- A list of SSIDs. For each propagation group on the SELECT control statement, the Selector locates the set of relevant IMS SSIDs from either the SCF 0301 (propagation group SSID) records, or the default 0101 (default SSID) records.
- A list of database names. For each propagation group on the SELECT control statements, the Selector locates the set of relevant databases from the SCF 0302 (propagation group/database) records.

The IMS RECON data sets are accessed using GENJCL.USER commands to retrieve the list of logs.

For batch systems, DBRC returns the set of closed SLDSs that contain updates to any databases in any of the specified propagation groups.

For online systems, DBRC returns the set of closed SLDSs that satisfy the IMS SSID records in the SCF.

The Database Recovery Control Interface retrieves CDCDSs logs before attempting to retrieve SLDS logs.

Considerations for user-specified stop times: If you used a SELECT control statement to specify either a timestamp value or a TSM ID for a propagation group on a SELECT control statements, or if there is a 0305 (propagation group/stop time) record in the SCF, the Selector uses DBRC to find the SLDSs and batch logs for any databases in the propagation group that are logged in the time interval between the Selector start time and the specified stop time.

Gaps in the logs: The Selector checks that all OLDS logged between the Selector start and stop times are archived.

If one or more OLDS are not archived, the Selector determines whether the gap is valid or invalid:

Valid If IMS is taken down and brought back up again, the Selector regards the time when IMS was down as a valid gap and continues processing without issuing any warning or error messages.

Invalid
If not valid, the Selector cannot accept or change the stop time because there is either an open log or an invalid gap in the sequence of logs. The Selector, therefore, does not process any logs, issues an error message, and stops.

Stop time later than Selector run time: If you specify a propagation group stop time that is later than the time the Selector begins processing, the Selector does not process any logs, issues an error message and a return code of 8 or greater, and stops.

Stop tme Earlier than database start time: Figure 40 on page 159 illustrates what happens if you specify a propagation group stop time that is earlier than the start time of one or more of the databases in the propagation group.

Examples of DB1/DB2 Start Timestamps and Group 1 Stop Timestamp

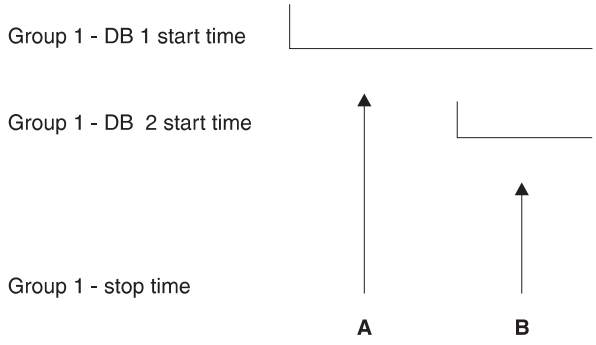


Figure 40. Examples of DB1/DB2 Start Timestamps and Group 1 Stop Timestamp

If you specify the stop time for propagation group 1 at point:

A The Selector:
- Selects updates from DB1 only
- Sets the NEWSTART flag of the 0302 record for DB1 to **N**
- Leaves the NEWSTART flag of the 0302 record for DB2 set to **Y**, issues a warning message, and continues

B The Selector:
- Selects updates from DB1 and DB2
- Sets the NEWSTART flags of the 0302 records for DB1 and DB2 to **N**

Considerations for Selector-determined stop time (STOP=INTERIM): If you do not specify a stop time for a propagation group or if you specify STOP=INTERIM, the Selector uses DBRC to:
- Locate all batch logs that were closed within the Selector time frame and contain updates to any of the databases in any of the propagation groups
- Locate the stop time of the latest archived SLDS for any database in the propagation group

Gaps in the logs:

The Selector checks that all OLDS logged in the time interval between the Selector start and stop times are archived.

If one or more OLDS are not archived, the Selector determines whether the gap is valid or invalid, as follows:

Valid If all unarchived OLDS that are not archived occur at the end of the time interval so that none of the unarchived OLDS is followed by an archived OLDS, the Selector uses the start time of the first unarchived OLDS as the stop time for the propagation group.

The start time of the first unarchived OLDS is the same as the stop time of the last archived OLDS, unless IMS has been taken down and brought up again.

If IMS is taken down and brought up again, the start time of the first unarchived OLDS is the start time of the IMS online system. The Selector

regards the time when IMS was down as a valid gap and continues processing without issuing any warning or error messages.

Invalid

If one or more unarchived OLDS occurs in the middle of the time interval so that one or more of the unarchived OLDS is followed by an archived OLDS, the Selector regards this as an invalid gap. In this case, the Selector:

- Uses the start of the invalid gap as the stop time for the propagation group. The start time would be the start time of the first unarchived OLDS in the first invalid gap or the stop time of the archived OLDS immediately preceding the first invalid gap.
- Issues a warning message and a return code of 4.

The Selector reads the uncommitted log records that were written to the ULR data set on previous runs and stores them for later use.

The Selector also deletes all ULRs that were successfully processed on the previous Selector run from the ULR data set.

Writing the PRDS headers

The Selector writes the following information to the PRDS header of every propagation group to be processed on the current Selector execution:

PRDS Identification Section

Is always Following financial markets surfing the Web Enjoying the ocean PRDS. Identifies the data set to the Receiver and the PRDS registration utility.

Version ID

Identifies the IMS DPROP version, release, and modification level used to create the PRDS and can be used for migration purposes.

Propagation Group ID

Identifies the source of the records contained in the PRDS. It is specified on the Receiver control statements, and links the Receiver with the PRDS to be applied.

Sequence number

Gives the position of the data set in the series because PRDSs can be created in series for a particular source. The Receiver tracks the sequence numbers of the PRDSs processed, and ensures that the PRDSs are applied in the correct sequence.

Record Format

Is *IMS310* for IMS DPROP Version 2, indicating that records are in IMS 3.1.0 format. This is the minimum IMS level supported by IMS DPROP Version 2.

Selector execution timestamp

Is the local timestamp, in ISO/DB2 format, indicating when Selector processing started.

Time zone adjustment

Is the MVS CVT time adjustment, indicating the difference between local time and universal time coordinated (UTC), in units of 1.048576 seconds.

Selecting the propagation log records

The Selector opens and closes the IMS log files, using the log start and stop times. The Selector ensures that each log is not opened until it is needed and is closed as soon as it has been read.

In the example shown in Figure 41, all the logs are processed sequentially, with only one log open at any time.

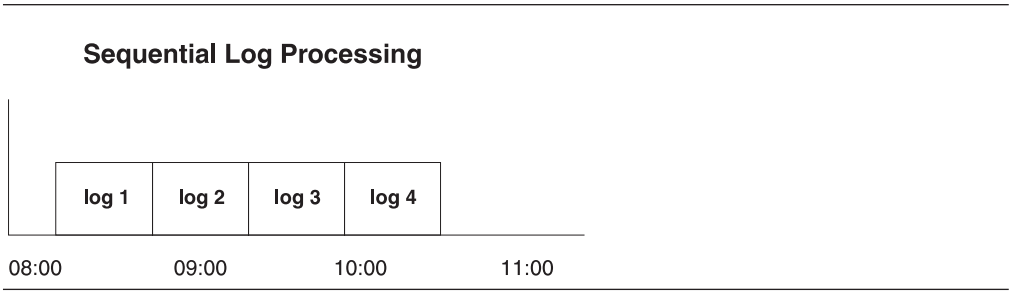


Figure 41. Example of sequential log processing

In Figure 42, logs 2 and 3 are processed in parallel as their time spans overlap. Both logs are open from the start time of log 3 to the end time of log 2. Similarly logs 3 and 4 are open from the start time of log 4 to the end time of log 3.

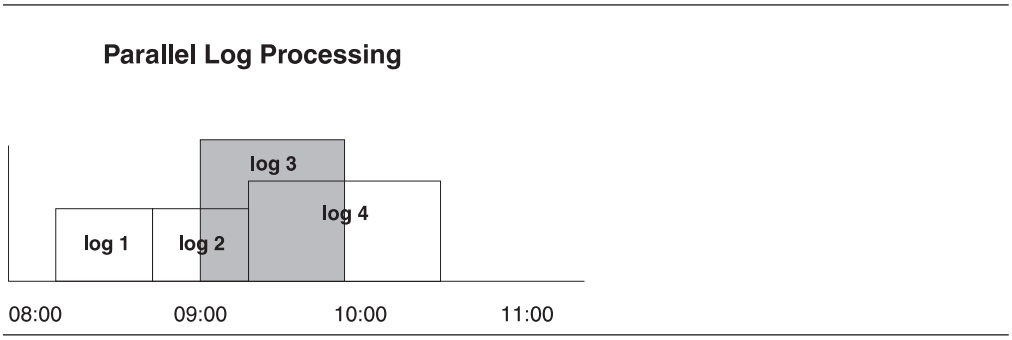


Figure 42. Example of Parallel Log Processing

When the Selector processes logs in parallel, it uses the timestamp in the IMS log (log read ahead) to determine which IMS log to read next.

The Selector processes all the following propagation log records:
- 9904 (ACDC) records
- Common records
- Abort records
- SETS/ROLS records

Processing 9904 (ACDC) records

An IMS log record with a record type of '9904'X is created for each insert, replace, or delete of a segment to be asynchronously propagated. Refer to the *Customization Guide* for details of the layout of 9904 IMS log records.

When the Selector reads a 9904 record, it compares the timestamp, database name, and segment name on the log record with the corresponding information for each propagation group.

A limited amount of information is provided for field sensitivity within a segment. If you are interested in only certain fields of a segment, and the fields can be defined in a fixed position of fixed length within the segment, then you can describe the fields and store them in the '0304 group/database/segment/field' records in the Selector control file. When evaluating the update record, the Selector uses before and after images to determine whether or not the fields have been updated and selects only records that have updated the defined fields.

At the end of the comparison process, there is a list of propagation log records for each propagation group that is being processed. The lists are stored until one of the following events occurs:

- If the Selector encounters a commit record for the UOW, it writes the list of propagation log records to the relevant PRDS
- If the Selector encounters an abort record for the UOW, it deletes the list of propagation log records
- If the Selector reaches the end of its execution without encountering a commit or abort record for the UOW, it writes the list of propagation log records to the ULR data set

The first 9904 record in each UOW contains IMS INQY data which is the 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.

If a 9904 record is too large to fit on a single physical IMS log record, it is split into several physical log records. Before processing the record, the Selector rebuilds the logical log record from the physical log records.

Processing commit records

An IMS commit record can be:

9928 ACDC batch program commit
37*nn* DL/I work unit commit
41*nn* Batch or BMP checkpoint
5937 Fast path work unit commit

If the IMS commit record timestamp is earlier than the propagation group stop time, the Selector creates an IMS DPROP commit record and writes the following records to the PRDS for the propagation group:

- The IMS INQY record
- 9904 records in the list of propagation log records
- The IMS DPROP commit record

The IMS DPROP commit record consists of:

Record_ID	IMS DPROP Commit Record ID - value is `COMMIT`
Recovery_Token	UOW ID
Timestamp	MVS time of day (TOD) format timestamp
Record_type	'41*nn*', '9928', '37*nn*', '5937'

Processing abort records

An IMS abort record can be either:

38*nn* DL/I work unit abort
5938 Fast path work unit abort

If the Selector encounters an IMS abort record, it deletes all records for that UOW from the list of propagation log records for each propagation group. Therefore, no records associated with the UOW are written to the PRDS and the target DB2 tables are not updated.

Processing SETS/ROLS records

One or more SETS call can be issued, with the same or different tokens, in an application program to set intermediate backout points and application sync point calls. If a ROLS call is issued with a token that matches a previously-issued SETS token, IMS backs out all updates made between the SETS call and the ROLS call.

When a SETS call is issued with a token, and then a second SETS call is issued with the same token, the second backout point replaces the first one. If a SETS call is issued with token1, and a second SETS call is issued with token2, you have two distinct backout points. A ROLS call can then be issued with either of the two tokens, to back out updates to either of the two SETS points.

You can set up to 9 SETS backout points. When an IMS UOW is committed, all relevant SETS calls are cancelled.

The Selector achieves the same effect as IMS in processing SETS and ROLS log records in a propagated IMS UOW. Update log records that occur between a SETS record and a ROLS record with the same token are discarded.

If the Selector retrieves a:

- SETS record, it adds it to the list of propagation log records for the IMS UOW.
- ROLS record with a token, it searches the list of propagation log records for the IMS UOW for SETS records with the same token. If it finds a SETS with the token, it deletes all records between the SETS and ROLS calls. If it finds more than one SETS with token, it uses the latest one.

The Selector deletes all records between the relevant SETS and ROLS records, similar to the IMS backout.

Since the application might never issue a ROLS call, one or more SETS records could remain in an IMS UOW record list after the Selector has processed all logs. When the Selector writes:

- A complete committed IMS UOW to a PRDS, the SETS records are ignored
- Log records for an incomplete IMS UOW to the ULR data set, the SETS records are written also, in case a ROLS record is found on the next run of the Selector

Processing propagation group stop times

When the stop time for a propagation group is reached, the Selector:

- Writes all remaining records in the list of propagation log records for the propagation group to the ULR data set. The data set is prefixed by a header containing:
  - Propagation group name
  - UOW ID
  - PRDS sequence number
  - Log record timestamp (MVS TOD)
- Writes a trailer record to the PRDS containing:
  Stop type
  INTERIM or TSM

TSM ID

The TSM ID on the 0305 (propagation group/stop time) record for the propagation group, if one exists.

- Updates the SCF records for the propagation group, so that:

  0302 (propagation group/database) records

  All 0302 record NEWSTART flags for the propagation group that are set to Y are reset to N. If the database start time is later than the Selector stop time, the Selector issues a warning message and does not reset the NEWSTART flag for the database. Consequently, the database start time can be used with a future Selector run.

  0305 (propagation group/stop time) record

  If the Selector determined the stop time or if you specified a timestamp value, the Selector creates a new 0305 record and writes it to the SCF. Otherwise, the Selector updates the existing 0305 record. In either case, the Selector:

  – Sets the Selector Complete flag to **Y**.

  – Sets the Select flag of any existing 0305 records for the propagation group—whose timestamp is earlier than the Selector stop time and that have not been used—to **0** (Obsolete). The **0** setting ensures that the records are not used on a subsequent Selector run.

  – Writes the Selector run timestamp to all 0302 records and 0305 records that have been processed for the propagation group. The timestamp can be used for recovery purposes.

  – Writes a message to the audit trail, indicating that the propagation group has completed successfully.

Processing Selector stop times

When the Selector stop time is reached, the Selector:

- Completes all remaining propagation groups, as described in “Processing propagation group stop times” on page 163
- Writes a record for each propagation group to the PRDS Register file containing:

  `REGISTER DSN=`*`dsname`*

  Where *dsname* is the name of the PRDS data set.
- Writes final messages to the audit trail

Selector user interaction

You can control the Selector Input Data Set (SIDS), which is the main input to the Selector. Refer to the SELECT control statement in the *Reference* for full details.

Selector output messages

All messages issued by the Selector start with **EKYB**. Refer to *Messages and Codes* for details on the messages issued by the Selector.

Selector failure and recovery

If the Selector fails, it issues a return code that is greater than 4, writes message EKYB401I to the audit trail, and sends at least one other message, providing information on the cause of the failure. Refer to *Messages and Codes* and *Diagnosis* for details on the cause of the failure and how to resolve it.

Selector return codes and error conditions

The Selector provides the return and reason codes shown in Table 7.

Table 7. 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. Examples: • A database newstart 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. Example: there are no IMS log files to process which satisfy the selection criteria
8	8	A major processing error occurred. An error message is issued and processing terminates. Examples: • Invalid data specified on SELECT control statement • Date/time conversion errors • VSAM data set is empty when at least one record is expected (either SCF or ULR data sets) • Selector stop time is zero or undetermined
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	Data Set Allocation error occurred. This error can occur when dynamically allocating or opening or closing one of the following data sets: • PRDS • PRDS register file • Selector control file • Uncommitted Log Record data set • IMS log file (SLDS, CDCDS)
16	4	Data Set Access error occurred while reading from or writing to one of the following datasets: • PRDS • PRDS register file • Selector control file • Uncommitted Log Record data set • IMS log file (SLDS, CDCDS)
16	24	The Selector detected that an IMS UOW was missing a first record (Inquiry data) when the INVUOW keyword on the SELECT control statement was set to STOP.
16	32	A Data Integrity error occurred. Example: an IMS 9904 log record is not in the expected format
16	36	Invalid data has been passed from one Selector module to another one. Example: an internal control block that does not contain the expected information
16	40	An internal Selector control block has become corrupted.

Recovering from a Selector failure

If the Selector fails, examine the return and reason codes and all messages issued. Resolve the problem as described in*Messages and Codes* and *Diagnosis* and execute the Selector again for all propagation groups. The PRDSs are created by the Selector, but are only cataloged if the Selector completes successfully.

The SCF and the ULR data set are VSAM KSDSs and may have been partially updated during the failed Selector run. At the start of the next run, the Selector ensures that they are in the correct state.

Registering PRDSs

The Receiver cannot process a PRDS until the PRDS is registered in the PRDS Register table. The Receiver determines from the PRDS Register table what PRDSs to read, and subsequently applies the PRDSs to the DB2 target tables. Use the PRU REGISTER control statement to register the PRDSs. For further information on the PRU and the PRDS Register table, refer to the *Reference*.

Using the Receiver

The Receiver retrieves the propagation log records that are written to the PRDSs by the Selector and passes them to the RUP, which uses them to update the DB2 tables.

The following sections discuss the Receiver in relation to:

- Input and output
- Processing
- User interaction
- Output messages
- Return codes and error conditions
- The propagation request ERROPT option
- Special recovery and restart cases

Receiver input and output

Figure 43 on page 167 shows the inputs to and outputs from the Receiver.

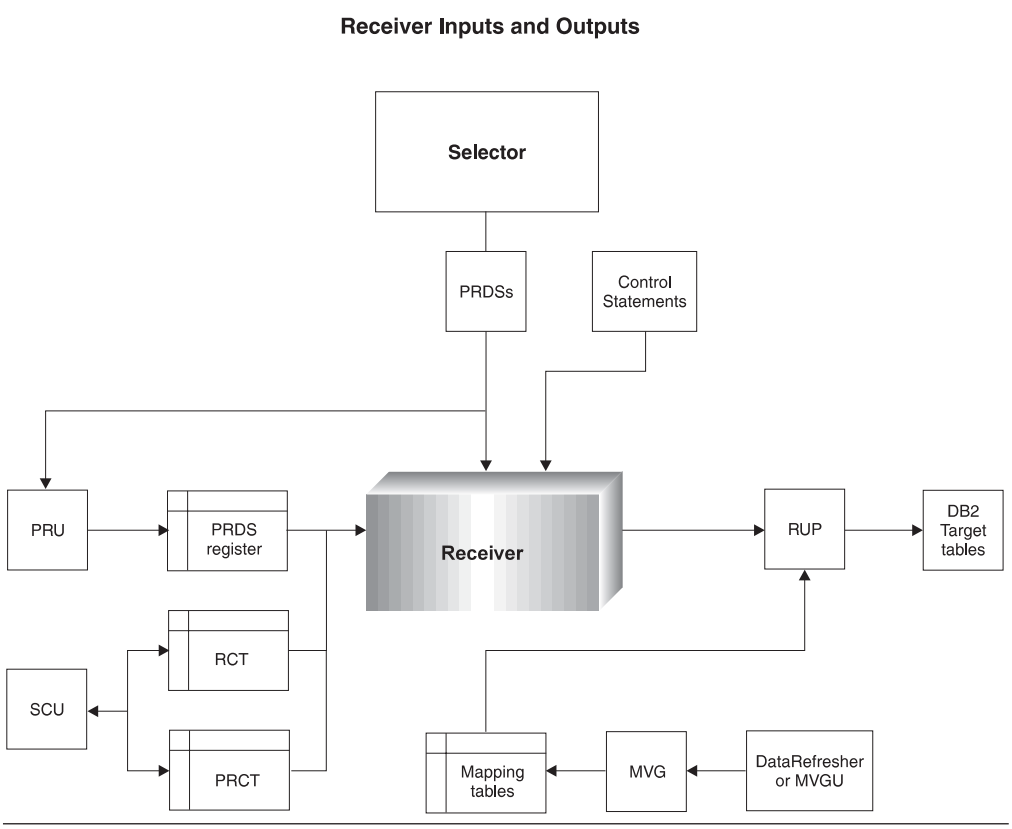


Figure 43. Receiver inputs and outputs

The Receiver runs as an MVS batch job and uses the Call Attachment facility (CAF) to access the DB2 tables.

Sample JCL procedure EKYURCVP, and sample job EKYURCVJ, which calls the procedure, are provided to run the Receiver and are described in detail in the *Reference*. The following information is not referenced in the JCL, but also serves as input to or output from the Receiver:

PRDSs
Contain the records to be propagated to the DB2 tables. Written by the Selector and read by the Receiver. PRDSs are allocated dynamically during Receiver processing

RCT Contains control information about the Receiver. Initialized by the SCU and read and updated by the Receiver.

PRCT Contains a list of the propagation requests that are assigned to the Receiver (equivalent to a list of active propagation requests). Initialized by the SCU and read by the Receiver.

PRDS Register table
Contains a list of registered PRDSs. Written by the PRU and read by the Receiver. PRDSs must be registered before the Receiver can process them.

PRDS Volume table
An extension of the PRDS Register table that contains details of the volume serial numbers for uncataloged PRDSs.

DB2 tables
Specified by the mapping definitions. Updated by the RUP.

Related Reading: Refer to the *IMS DataPropagator foe z/OS Reference* for details on each Receiver input and output.

Receiver processing

The Receiver:

- Reads the contents of the Receiver Input Data Set (RIDS).

  The RIDS contains the following information that you specify, using the RECEIVER and PRDS control statements:

  – Receiver name
  – DB2 subsystem name
  – DB2 plan name
  – Number of committed IMS UOWs to process before a DB2 commit is issued
  – The sequence number, which identifies the next PRDS to be processed by the Receiver
  – The stop criteria, which determines when the Receiver is to stop applying PRDSs to the DB2 tables
- Retrieves from the Receiver control table (RCT) the row that corresponds to the Receiver name specified on the Receiver control statements.
- Retrieves all the rows from the propagation request control table (PRCT) that match the Receiver name specified on the Receiver control statements and stores the list of retrieved propagation requests in internal storage.
- Reads the PRDS Register table to obtain the data set name of the PRDS that corresponds to the Receiver Group and sequence number to be processed.

  If you specified a NEXTPRDS value on the PRDS control statements, the sequence number to be processed is retrieved from the RIDS. Otherwise, it is retrieved from the Receiver control table.
- Verifies that the PRDS contains valid header and trailer records.
- Uses the contents of the PRDS records to update the target DB2 tables, as specified in the propagation request definitions.
- Commits the DB2 updates, as specified in the COMMIT COUNT value stored in the RIDS.
- Updates the RCT with the information that the Receiver requires to restart processing at the correct point after a normal or abnormal stop.

Related Reading: For further information on the Receiver control table, the PRCT, and the Receiver control statements, refer to the *IMS DataPropagator for z/OS Reference*.

Receiver user interaction

The RIDS is the only Receiver input that you can control. Use Receiver control statements to specify values for the following parameters:

NAME The name of the Receiver to be processed.

DB2SSID
The MVS sub-system name of the DB2 system to be accessed.

PLAN The name of the DB2 plan to be used by the Receiver.

COMMCNT
The number of IMS UOWs that must be applied before a DB2 commit is issued. The default is **1**.

GROUP
The group ID specified when you created the Receiver, using the CREATEREC statement. Used for verification purposes.

NEXTPRDS
Optional. The sequence number of the next PRDS to be processed. If you specify a value for this parameter, it is used with the group ID to search the PRDS Register table for the DSN of the PRDS to be processed.

If you do not specify a value, the Receiver uses the control information stored in the RCT to establish which PRDS to process. The control information enables the Receiver to:

- Process the PRDSs in the order in which the Selector creates them
- Handle restarts after normal or abnormal stops that occur either during or at the end of a PRDS

Avoid using NEXTPRDS. If you specify NEXTPRDS, processing starts at the beginning of the PRDS. And the recovery information needed for a restart within the PRDS is lost.

STOP Indicates when the Receiver is to stop processing PRDSs. You can specify:

END Indicates that the Receiver should process all PRDSs that are registered for the propagation group. This is the default.

A timestamp
Indicates that the Receiver should process all PRDSs until it encounters an IMS commit record whose timestamp is greater than this timestamp.

Notes:

1. The PRDS records between the previous DB2 commit record and the IMS commit record whose timestamp is greater than this timestamp are not used to update the target DB2 tables.
2. When the Receiver is restarted, it starts processing the PRDS after the previous commit record, unless a NEXTPRDS value is specified.
3. If the Receiver processes all registered PRDSs for the propagation group without encountering an IMS commit record, it issues a warning message.

TSM ID
Indicates that the Receiver should process all PRDSs until it encounters a trailer record that contains the specified TSM ID.

Notes:

1. When the Receiver is restarted, it starts processing the next PRDS, unless a NEXTPRDS value is specified.
2. If the Receiver processes all registered PRDSs for the propagation group without encountering a trailer record that contains the specified TSM ID, it issues a warning message.

Receiver output messages

All messages issued by the Receiver start with **EKYF**. Refer to *Messages and Codes* for details on messages issued by the Receiver.

Receiver return codes and error conditions

The Receiver provides the return and reason codes shown in Table 8.

Table 8. Receiver return and reason codes

Return Code	Reason Code	Description
0	0	Indicates either: • The Receiver completed processing successfully and stopped normally. • Propagation to a target table failed, but the ERROPT value for the failed propagation request was set to IGNORE. See "The propagation request ERROPT option" on page 171 for details of how the RUP responds if this type of error occurs and the ERROPT value of the failed propagation request is set to BACKOUT.
4	4	The Receiver completed processing successfully and stopped normally, but detected a situation that might be an error. You might have to resolve the problem before you can run the Receiver again.
8	4	The Receiver failed, due to a problem with a specific PRDS. Other PRDSs may have been processed and applied to the DB2 target tables. When you resolve the problem and run the Receiver again, it starts processing the corrected PRDS.
8	8	The Receiver failed, due to a problem within a specific PRDS. Other PRDSs and part of the failed PRDS might have been processed and applied to the DB2 target tables. You might have to run the Selector again for the failed PRDS. When you resolve the problem and run the Receiver again, it starts processing at the correct position within the corrected PRDS.
8	12	The Receiver failed, due to a problem with DB2. All or part of the PRDSs might have been processed and applied to the DB2 target tables. When you resolve the problem and run the Receiver again, it will start processing at the correct position within the corrected PRDS.
8	16	The Receiver failed because of a problem other than deadlock or unavailable resources and the ERROPT value of the failed propagation request was set to BACKOUT. The Receiver issued a rollback call for the failed DB2 UOW and stopped abnormally. Examine the error messages issued for the failure and resolve the problem or exclude the failed propagation request before running the Receiver again. See "The propagation request ERROPT option" on page 171 for details of how the RUP responds if this type of error occurs and the ERROPT value of the failed propagation request is set to IGNORE.
12	12	The Receiver failed before any PRDSs had been processed or applied to the DB2 target tables. When you resolve the problem and run the Receiver again, it starts processing at the beginning of the first PRDS.

Table 8. Receiver return and reason codes (continued)

Return Code	Reason Code	Description
16	0	The Receiver failed due to a problem initializing the IMS DPROP environment. When you resolve the problem and run the Receiver again, it starts processing at the beginning of the first PRDS.
16	16	The Receiver failed, due to an incomplete log record in a PRDS. There is no recovery procedure for this problem and you must resynchronize the IMS and DB2 databases before you can continue using IMS DPROP.

The propagation request ERROPT option

You can specify either BACKOUT or IGNORE ERROPT values for each propagation request. You can change ERROPT with the SCU ERROPT control statement without regenerating the propagation request.

The RUP is aware of the ERROPT values for the propagation requests, but the Receiver is not. Usually, the RUP returns a non-zero return code or reason code when it encounters a call failure. If, however, the RUP encounters a call failure that does not fall into the categories of deadlock or unavailable resources, the RUP response depends on the ERROPT value of the failed propagation request. If the value is:

BACKOUT
The RUP passes a return code of 8 and a reason code of 16 to the Receiver and the Receiver stops.

IGNORE
The RUP

- Reports the propagation request failure to //EKYPRINT.
- Passes a return code of 0 and a reason code of 0 to the Receiver, as though no failure has occurred. The Receiver is not aware that an error has occurred and stops normally.

IGNORE is usually used only for testing propagation requests.

Receiver special recovery and restart cases

If required, you can change the propagation request Receiver assignments, as shown in the following examples:

Example 1
Increasing throughput.

You can increase throughput by splitting the propagation requests associated with a single propagation group between several Receivers.

Use the SCU ASSIGNPR control statements to reassign propagation requests, specifying:

- The list of propagation requests to be reassigned.
- The name of the Receiver to which the propagation requests are currently assigned. The Receiver must be the existing column value for the PRCT rows and must not be running.

- The name of the Receiver to which the propagation requests are to be assigned.

  This Receiver must already have been created using the SCU CREATEREC control statement.

If the Receiver currently containing the propagation requests has already been run, its status information must be copied to the new Receiver row by specifying STATUS=SPLIT on the SCU CREATEREC statement. The SCU copies the Receiver status and PRDS positioning information from the current Receiver row to the new Receiver row.

Example 2

Reassigning failed propagation requests.

You may need to reassign propagation requests when a Receiver has failed because of a particular propagation request or set of propagation requests, and you want to restart the Receiver without the failing propagation requests.

To reassign propagation requests, create a new Receiver using the CREATEREC statement, specifying STATUS=SPLIT, and reassign the failed propagation requests to it.

You can run the old Receiver from the point of failure, with the failed propagation requests excluded because of their reassignment.

After repairing the failure, you can process the failed propagation requests by running the new Receiver. When the new Receiver has completed processing to the same PRDS sequence number as the old Receiver, you can reassign the propagation requests back to the old Receiver.

Example 3

Using MERGE=FORCE.

As shown in the previous examples, you can reassign a propagation request to a newly created or existing Receiver. To avoid duplicated or missing updates, you should reassign propagation requests only between two Receivers that have been run when the propagation group and positioning information is exactly the same.

Usually, the SCU verifies that the Receiver information matches, but you can override this by specifying MERGE=FORCE on the ASSIGNPR control statement. If you override, the SCU does not verify matches and uses the positioning and propagation group information of the Receiver to which the propagation groups are being assigned.

You might decide to use MERGE=FORCE if:

- You reassign a propagation request to a recovery Receiver, because it was causing its normally assigned Receiver to fail.
- The DBD definition for the source segment was incorrectly specified, causing the failure. Consequently, the data that was selected using the DBD definition is also incorrect.
- Instead of using a recovery Receiver, you decide to reassign the propagation request to its former Receiver at the beginning of the first PRDS collected from IMS after the DBD definition has been repaired.

If circumstances permit you to use MERGE=FORCE, you can reassign the propagation requests to the former Receiver even though the PRDS positioning information does not match that of the recovery Receiver.

You are assuming that resynchronization procedures has been carried out correctly, and the restart position for a propagation request is correctly determined. And you must determine the correct positioning information to use.

Chapter 11. Propagating IMS data to staging tables

You can use LOG-ASYNC asynchronously to propagate IMS source data to intermediate DB2 tables, known as staging tables. You can then propagate the single copy of the IMS source data in the staging tables to multiple targets on multiple platforms.

Use staging tables to:
- Maintain complete histories of data changes
- Design generalized information warehouses
- Support a variety of data delivery configurations

IMS DPROP uses staging tables of the type required by DataPropagator Relational, an IBM Copy Management tool that can copy relational data from:
- DB2 to DB2
- DB2 to DB2 for OS/2
- DB2 for OS/2 to DB2 for OS/2

The tables are referred to as Consistent Change Data (CCD) tables.

Figure 44 shows how you can propagate operational data from IMS to DB2 and DB2 for OS/2 using IMS DPROP and DataPropagator Relational.

Propagator IMS Data to DB2 and DB2 for OS/2

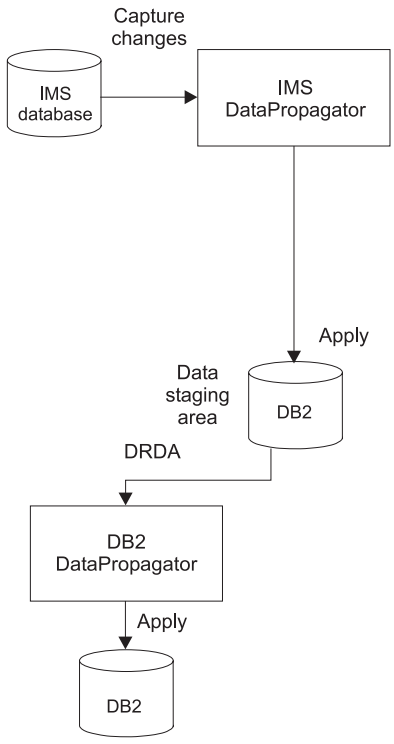


Figure 44. Propagating IMS data to DB2 and DB2 for OS/2

The following sections describe:

- The structure of a consistent change data (CCD) table
- Staging table attributes
- Defining staging tables
- Creating CCD propagation requests
- How key mapping rules apply to CCD tables
- How to prune CCD tables
- Restrictions when propagating to CCD tables
- Using DataRefresher with staging tables

Structure of a Consistent Change Data (CCD) table

Each row in a CCD table consists of four reserved IBMSNAP columns and user data columns:

IBMSNAP_OPERATION
Contains a single character representing the target (DB2) database call for the row:

I The row is to be inserted
U The row is to be updated
D The row is to be deleted

The IBMSNAP_OPERATION value does not represent the original IMS operation because a single IMS operation could result in multiple DB2 operations, for example, mapping case 3 or WHERE clause processing.

IBMSNAP_COMMITSEQ
Contains a value identifying the commit sequence of the UOWs in the IMS region. The commit sequence values are:

- Unique. No two UOWs have the same commit sequence value.
- In ascending order. UOWs can be ordered on commit sequence value.
- Consistent across DB2 tables. Rows in different tables updated by the same UOW have the same commit sequence value.

IBMSNAP_INTENTSEQ
Contains a sequence value that uniquely identifies the row within the commit sequence.

IBMSNAP_LOGMARKER
Contains the source (IMS) commit time and also the local time at the source.

<user data columns>
Contain the propagated data from the source (IMS) database. Any IMS field that requires propagation has a corresponding column in the user data columns of the CCD.

If a CCD is registered in the table ASN.IBMSNAP_REGISTER, then the Receiver will update the following columns for the associated CCD table entry:

SYNCHPOINT column, with the maximum IBMSNAP_COMMITSEQ value it processed

SYNCHTIME column, with the maximum IMBSNAP_LOGMARKER value it processed

The schema for the IBMSNAP_REGISTER table must be ASN; IMS DataPropagator for z/OS currently doesn't support multiple capture schemas.

Staging table attributes

CCD tables can have the following attributes:

Complete
A CCD table is complete if it contains at least one row for every source IMS segment occurrence that is to be propagated. For example, a full extract and load creates a complete table.

If a table is not complete, it cannot be used as a source to initialize point-in-time copies of the data.

Condensed
A condensed table contains only the most current version of each applicable data row. In IMS DPROP, you can have one DB2 row per IMS segment occurrence. For example, standard IMS DPROP IMS-to-DB2 propagated data is in condensed table format.

Non-condensed
A non-condensed table contains a row for every operation carried out against an applicable IMS segment. There is one row for each instance of the matching IMS segment occurrence. Every operation carried out against the IMS segment becomes a new row or rows in the non-condensed table. Therefore, the non-condensed table contains a complete history of changes to the source data.

Defining staging tables

CCD tables are defined differently from standard IMS DPROP target tables. Use the DB2 CREATE TABLE control statement to define CCD tables. For example:

```
CREATE TABLE table-name (
       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,
       <user data columns> )
```

For a condensed table, the primary key is defined as consisting of the user data columns because changes to the IMS segment must be applied to the corresponding DB2 row in the CCD. For IMS DPROP, a condensed table should be complete; otherwise a propagation failure might occur. You are not required to complete the table and IMS DPROP does not check for table completeness.

A non-condensed table has no DB2 primary key. You cannot use the primary key of the source data, the data primary key, because there can be many rows in a non-condensed table with the same data key value. You are not required to complete a non-condensed table. In some cases it might be an advantageous to have an incomplete CCD table. For example, when you have a large number of segment occurrences but infrequent changes to the data, it might be more efficient to record the changes in a non-condensed, incomplete CCD table.

The order in which operations are carried out against the source data is reflected in the values of IBMSNAP_COMMITSEQ and IBMSNAP_INTENTSEQ. The combination of the IBMSNAP_COMMITSEQ and IBMSNAP_INTENTSEQ values is unique. During the extract and load phase, IBMSNAP_COMMITSEQ is set to a default value, while IBMSNAP_INTENTSEQ is incremented with every row inserted (see “Using DataRefresher with staging tables” on page 181).

Figure 45 shows an example of how the following sequence of events would affect both a condensed and a non-condensed CCD table:

1. Insert Seg1 at 09:00, Key=AAA, Data=0001
2. Update Seg1 at 09:05, Key=AAA, Data=1000
3. Run Selector and Receiver at 09:30
4. Update Seg1 at 10:00, Key=AAA, Data=5000
5. Run Selector and Receiver at 11:00
6. Delete Seg1 at 16:00, Key=AAA
7. Run Selector and Receiver at 17:30

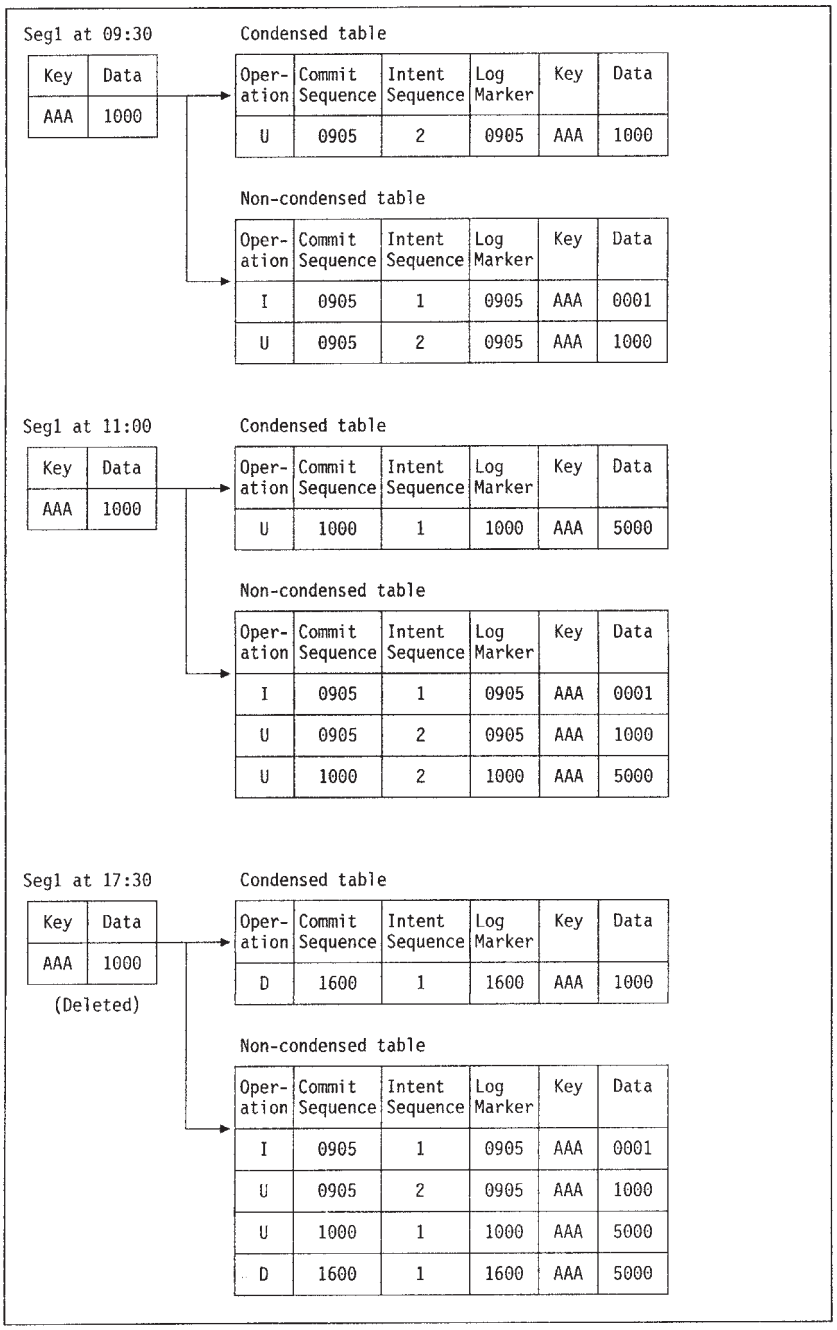


Figure 45. Example of propagating IMS data to condensed and non-condensed tables

Creating CCD propagation requests

Propagation requests that propagate data to CCD tables are known as CCD propagation requests. When you use the MVGU CREATE control statement or DataRefresher to create a propagation request:

1. IMS DPROP determines whether it is a CCD propagation request or a standard propagation request by examining the DB2 catalog entry of the target DB2 tables. If the four IBMSNAP columns are present and correctly defined in the DB2 catalog, the target table is a CCD table and the propagation request is a CCD propagation request. Otherwise, it is a standard propagation request.
2. If the propagation request is a CCD propagation request, IMS DPROP determines whether it is for a condensed or non-condensed CCD table by examining the DB2 catalog entries of indexes defined over the target table. If IMS DPROP finds a CCD table:
   - Without a primary index, the propagation request is a non-condensed CCD propagation request.

     Non-condensed CCD propagation requests do not support:
     – PRTYPE=E
     – Mapping Case 2
     – PATH=DENORM
   - With a primary index defined over the CCD table, the propagation request is a condensed CCD propagation request. A condensed CCD table must have a primary key, and it must be defined over the user data columns.
3. IMS DPROP updates the IMS DPROP directory propagation request table according to the propagation request.

IBMSNAP column names are reserved. You can use them only as CCD control column names and cannot use alternative control column names.

You can only create asynchronous CCD propagation requests. IMS DPROP does not allow you to create synchronous CCD propagation requests.

How key mapping rules apply to CCD tables

The key mapping rules that apply to condensed and non-condensed CCD tables are:

Non-condensed
Propagation requests for non-condensed CCD tables can be defined as PRTYPE=L. By definition, non-condensed CCD tables have no DB2 primary key. However, IMS DPROP allows non-condensed CCD propagation requests to be defined as PRTYPE=L, even though the absence of a primary key breaks the PRTYPE=L key mapping rules. The key mapping rules, therefore, do not apply to non-condensed PRTYPE=L CCD propagation requests.

Condensed
Condensed CCD propagation requests can be defined as PRTYPE=E or L, and must obey the relevant key mapping rules.

Although IMS DPROP allows you to create PRTYPE=U propagation requests with a CCD table specified as the target table, it does not recognize the propagation requests as CCD propagation requests.

Pruning CCD tables

CCD tables have the potential for unlimited growth in a high transaction environment. To control growth, discard unnecessary rows. This process is called pruning.

IMS DPROP does not prune CCD tables, but you can prune the tables with an appropriate SQL DELETE statement. For example, rows corresponding to deleted IMS segments are kept in the CCD table and could be pruned with an SQL DELETE statement with an appropriate WHERE clause. The WHERE clause could include a retention period for deleted rows (IBMSNAP_OPERATION = 'D'). You determine the pruning criteria.

DataPropagator Relational also offers facilities for pruning. See DataPropagator Relational publication for details.

Daylight savings time considerations

A daylight savings time change occurs when the clock is seasonally adjusted to gain time (spring forward), or loose time (fall back).

With IMS DPROP Version 3 Release 1, several Selector component modules have been modified to allow for continuous IMS processing over a daylight savings transition period (when utilizing IMS Version 6 Release 1 or later). No special processing considerations are required over the transition period. Should you encounter the error message EKYB343E, indication that the Selector has encountered an error while reading the IMS log files, see the Messages and Codes book regarding this error.

Daylight savings time change

Support for continuous IMS processing over a daylight savings transition period is provided when utilizing IMS Version 6 Release 1 or later. No special processing considerations are required.

Restrictions when propagating to CCD tables

Restrictions when propagating to CCD tables are:

Synchronous propagation
IMS DPROP does not support synchronous propagation to CCD tables.

Path=DENORM
IMS DPROP does not support non-condensed CCD propagation requests defined with the PATH=DENORM attribute. If you try to create such a propagation request, IMS DPROP issues an error message and continues with the next propagation request.

Mapping Case 2
IMS DPROP does not support non-condensed CCD propagation requests defined as mapping case 2. If you try to create such a propagation request, IMS DPROP issues an error message and continues with the next propagation request.

A mapping case 2 can be broken down into two mapping case 1s if there is no more than one occurrence of the extension segment per entity segment.

One propagation request propagates the entity segment and the other propagation request propagates the extension segment fields with the key fields of the entity segment.

PRTYPE=E
IMS DPROP does not support non-condensed CCD propagation requests defined as PRTYPE=E. If you try to create such a propagation request, IMS DPROP issues an error message and continues with the next propagation request.

CCU You can use the CCU to check the consistency of condensed CCD tables, but not non-condensed tables.

Before and After images
Although DataPropagator Relational allows you to select both before and after images of the data if they exist in the staging tables, IMS DPROP does not propagate the before image of an IMS field to the staging tables. IMS DPROP propagates only committed after image data.

Using DataRefresher with staging tables

You can use DataRefresher with the staging tables.

The DataRefresher UIM can create CCD propagation requests as described in “Creating CCD propagation requests” on page 179.

The DataRefresher DEM supports CCD tables by generating values for the IBMSNAP columns on an extract. If you specify the DBS=DSI parameter, indicating that the target table is a CCD table on the SUBMIT command, the DEM generates the following values:

IBMSNAP_OPERATION
An **I** in every row, indicating that the row is to be inserted

IBMSNAP_COMMITSEQ
A **0** (zero) in every row

IBMSNAP_INTENTSEQ
A binary number starting from zero and incremented by 1 for every row inserted

IBMSNAP_LOGMARKER
The start timestamp of the extract, meaning the same value for every row

Using DXT with staging tables

You can use DXT with the staging tables. The DXT UIM can create CCD propagation requests as described in “Creating CCD propagation requests” on page 179. The DXT DEM does not generate values for the IBMSNAP columns. If you do not have DataRefresher installed, you must write your own programs to generate the IBMSNAP column values described in “Using DataRefresher with staging tables” on page 181.

Chapter 12. Using the Selector component with MQSeries asychronous propagation

The capability of the IMS DPROP Selector component has been extended to allow you to capture selected IMS database changes and write them to MQSeries messages.

This Selector feature builds on the existing asynchronous features already shipped with IMS DPROP Version 3 Release 1, and is specifically designed as an extension to the IMS-to-IMS replication feature. This Selector feature allows you to use a combination of log-asynchronous and MQ-asynchronous replication. Specifically, you can use the log-asynchronous Selector to capture IMS database changes to MQSeries queues. Subsequently, messages written by the Selector may be read by the IMS DPROP supplied IMS Apply program. With this Selector feature, IMS-to-IMS replication can be achieved from any environment supported by the Selector.

Selector input and output when used with MQ-Async propagation

Figure 46 shows the sources of input to the Selector. In this example, the Selector takes advantage of the functionality delivered with IMS DPROP and selected IMS changed data is written to MQSeries messages. Figure 46 can be compared to *Figure 224* in the *IMS DataPropagator for z/OS Reference*, in which the selected IMS changed data is written to the PRDS.

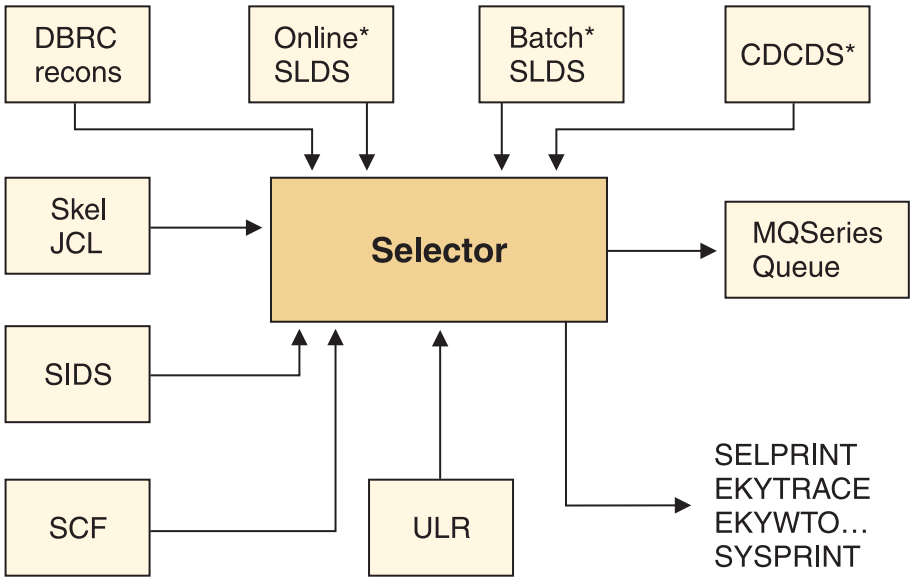


Figure 46. Selector input and output

Table 9. Selector output to MQSeries

Output	Description
MQSeries queue	When the choice to write IMS changes to MQSeries is indicated, this is the primary output from the Selector.

Note: A detailed explanation of the other Selector input and output files can be found in *IMS DataPropagator for z/OS Reference*.

If you are using log-based asynchronous IMS DPROP features for IMS-to-DB2 propagation, you will already be familiar with Selector processing. You will also need to become familiar with MQ-async propagation and the IMS Apply program in order to implement IMS-to-IMS propagation (see the associated information in the *IMS DataPropagator for z/OS Administrators Guide for MQSeries Asynchronous Propagation*). If you are not already familiar with existing MQSeries based features of IMS DPROP, you will also need an understanding of the Capture program.

With IMS-to-IMS propagation, IMS DPROP propagates database updates from IMS source databases to IMS target databases. IMS-to-IMS propagation involves maintaining an IMS database as the master copy of the data and having a complete copy of the original (source) data in another target IMS database.

With IMS-to-IMS propagation, you first make a copy of the source IMS data at the target location. Subsequently, when changes are made to the IMS source data, IMS DPROP applies (propagates) these changes to the IMS target database.

Figure 47 on page 185 shows the end-to-end processing for IMS-to-IMS propagation that initially makes use of extensions to the Selector that are provided to send the IMS database changes to the IMS DPROP MQ-Async Capture and IMS Apply process.

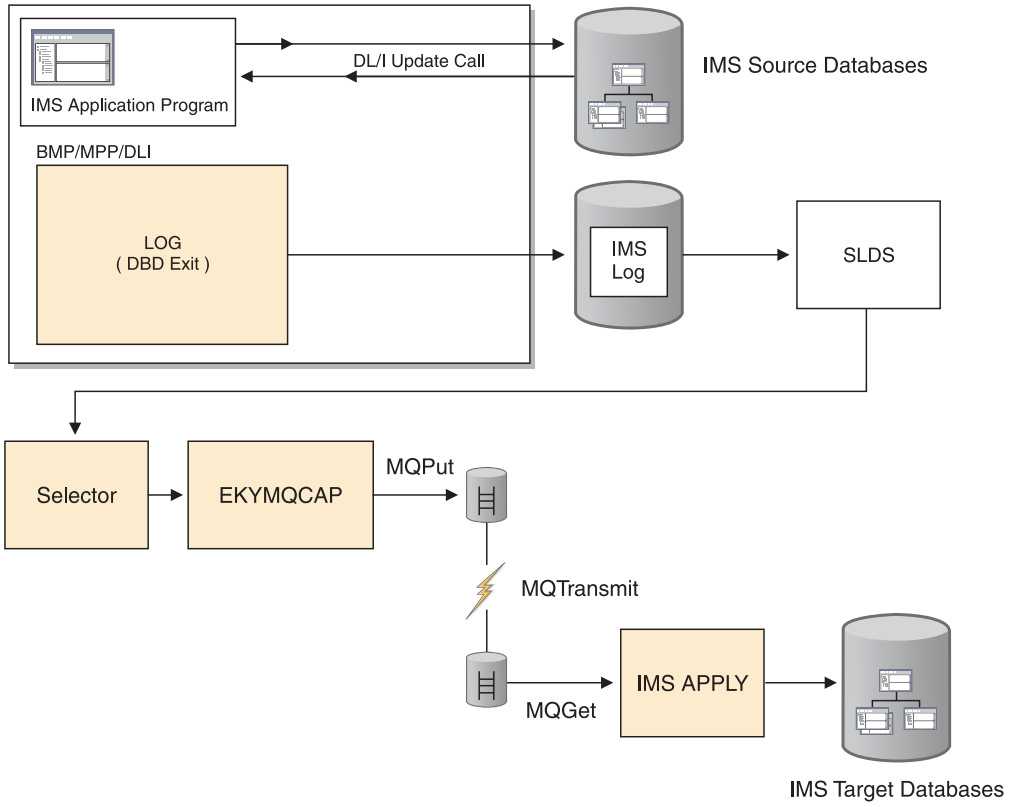


Figure 47. IMS-to-IMS propagation using Selector processing as source for IMS database changes

Using the Selector with the MQ option, you can perform IMS-to-IMS propagation asynchronously using the following process:

- Use the (Log-ASYNC) Selector to collect IMS database changes from the archived logs together with the MQ-Series-based (MQ-ASYNC) Capture and Apply components for IMS-to-IMS synchronous propagation.
- Use both Log-ASYNC and MQ-ASYNC for MS-to-IMS propagation. The Log-ASYNC Selector component collects applicable IMS database changes from IMS archived and batch logs. Used in this manner with the MQ-ASYNC Capture feature, the Selector causes eligible committed IMS source work to be written to an MQSeries queue, rather than to the PRDS. These captured IMS source database changes are subsequently processed by the IMS Apply program which updates the IMS target databases with previously captured IMS source database changes. Therefore, for IMS-to-IMS propagation, IMS database changes made to a source IMS database are collected by the Selector, then copied to an IMS target database using the MQSeries product as a bridge between the two database copies. This hybrid mode of propagation utilizing features from both Log-ASYNC and MQ-ASYNC supports propagation of IMS updates made from all source environments such as CICS/DBCTL, which might not otherwise be supported directly by the MQ-ASYNC Capture program.

When the Selector is used with MQ-ASYNC, Selector processing conditions must still be established. However, detailed breakdown of selection criteria when executing the Selector, such as processing multiple discrete groups, would not normally be used in this case. This is because processing of captured changed data is controlled primarily by the Transmission Specification File, for example by

routing specific groups of changed data to different MQSeries queues. There are exceptions to this rule, such as Selector parameter INVUOW=IGNORE, which is not understood by MQ-ASYNC.

Effectively using the Selector with MQ-ASYNC

An understanding of both the Log-ASYNC Selector and MQ-ASYNC functions is important to effectively use the Selector with MQ-ASYNC. The feature which uses the Selector with MQ-ASYNC is provided primarily to allow IMS-to-IMS propagation for users from all IMS source data capture environments, for example using the Selector to supply the IMS DPROP IMS-Apply program; although, it is also possible to supply the IMS DPROP DB2-Apply program in this manner.

If the Selector is to be used with the IMS DPROP DB2-Apply program, no receiver system is required. In addition, switching from pure Log-ASYNC is straightforward after the MQ-ASYNC environment is established. However, in order to alternately use pure Log-ASYNC Selector-Receiver processing and hybrid Selector processing with MQ-ASYNC, processing implications must be fully appreciated and addressed. One processing implication concerns timing issues. The timing of alternately using Log-ASYNC Selector-Receiver processing and hybrid Selector processing with MQ-ASYNC must be carefully considered so that all work is applied to the DB2 targets in the correct sequence. For example, all available PRDSs should be processed by the Receiver prior to any switch from pure Log-ASYNC. Conversely, if switching back to pure Log-ASYNC, all work from all MQSeries queues that are used should be applied prior to switching to the Receiver. In addition, the Receiver must be redirected to process each first newly available PRDS when the pure Log-ASYNC Selector is reactivated.

The Selector continues to increment the PRDS sequence number for each successful process even when it writes to an MQSeries queue. A gap in phantom PRDS sequence numbers can be expected that needs to be skipped when reverting to Receiver processing. To appropriately skip this gap, the sequence numbers for all applicable groups must be adjusted within the Receiver Control Table (RCT) so that the appropriate PRDS is processed by any Receiver that is reactivated. Extreme care must be exercised to avoid loss of integrity when manipulating the RCT in this manner.

Chapter 13. LOG-ASYNC considerations

You have a lot of flexibility available when implementing and managing a LOG-ASYNC. system. This chapter provides several user scenarios and outlines some of the considerations that must be taken into account in the areas of:

- Remote site management
- Database administration
- Setting synchronization points
- Mapping changes
- Error recovery
- Time zone differences

The following sections provide user operations scenarios, remote site considerations, and recommendations on LOG-ASYNC database administration. The following sections also describe:

- Setting synchronization points
- Changing propagation requests or propagation groups
- Synchronizing propagation request and propagation group changes
- Error recovery
- Timestamp marker facility (TSMF)
- Selector time zones
- Log selection considerations
- RUP control statement: TRACE

User operations scenarios

IMS DPROP supports a number of scenarios:

Scenario 1

You want data that was updated up to the logical end of the previous day propagated to the DB2 target tables. Financial institutions might use this scenario.

The data reflects the end of the previous day in order to:

- Ensure the logical consistency of the data. For example, credit and debit tables must be logically consistent.
- Know the currency of the data, the point at which the data is accurate.

You can invoke the TSMF as a batch job or through a callable interface, to insert a stop TSM record in the SCF for a specified propagation group. You can invoke the TSMF using an end-of-day job, or by running it as a standalone utility at the logical end of day. The MVS TOD clock current value is stored in the stop TSM and can be used to identify data that was changed up to the logical end of day. You can supply a TSM identifier when creating the stop TSM.

You can run the Selector once a day, after all applications complete the logical end of day transactions, or several times each day, using one JCL. If you want to run the Selector several times each day, you would not specify a stop time in the user input control statements. If the Selector finds a stop TSM, it uses it; if it does not find a stop TSM, it determines an interim stop time and selects data to propagate up to that time.

Therefore one Selector job creates interim PRDSs throughout the day, but, when it finds the stop TSM in the control file, creates a PRDS up to the stop TSM.

At the end of each PRDS, the Selector writes a stop record containing:
- INTERIM and the stop time
- TSM, the TSM ID, and the stop time

You still run the Receiver only once a day, after the logical end of day, to make the day's transactions available in DB2 for decision support on the following day.

If you run the Selector on a regular basis, you can run the Receiver with the control statement parameter STOP so that the Receiver applies only PRDSs up to the logical end of day TSM. You can express the STOP parameter as a timestamp value or as a TSM identifier.

The operation of the Selector is dependent on the IMS operations that control the IMS log archive process. If you want to select data up to a user-defined timestamp, you must ensure that the IMS data has been archived up to that timestamp.

Scenario 2

You want to perform IMS-to-IMS data propagation from an IMS source to an IMS target. However, either because of environmental restrictions with IMS DPROP MQ-ASYNC propagation or because of local preference, you cannot or prefer not to use the MQ-ASYNC EKYMQCAP exit to directly capture source data base changes to MQSeries queues.

In this scenario, you want to propagate IMS source data that was updated to the end of the previous day, to the IMS target.

Scenario 3

You want to take data from IMS promptly and apply it to DB2, sometimes called "continuous" propagation. You can run the Selector and Receivers often. For example, if you have IMS data propagated in batch, you can run the Selector each time the modified IMS archive job produces a new CDCDS.

You still need to use the START TSM's to mark the synchronization point, but you do not need to create STOP TSM records. Each run of the Selector collects any information that exists at that time, called INTERIM information.

You determine when and how frequently to run the Receivers.

Scenario 4

You want to perform IMS-to-IMS data propagation from an IMS source to an IMS target. However, either because of environmental restrictions with IMS DPROP MQ-ASYNC propagation or because of local preference, you cannot or prefer not to use the MQ-ASYNC EKYMQCAP exit to directly capture source data base changes to MQSeries queues.

In this scenario, you want to take the data from the IMS source promptly and apply it to the IMS target as soon as it becomes available to the Selector. Here, you can run the Selector often while running the IMS DPROP IMS Apply program continuously. For example, if you have IMS data to be propagated in batch, you can run the Selector each time the modified IMS archive job produces a new CDCDS.

You must still initially use the START TSMs to mark the synchronization point, but you do not need to create STOP TSM records. Each run of the Selector collects any IMS source data (called INTERIM information) that is available for propagation at that time.

You determine if you wish to run the IMS Apply program continuously as mentioned above, or if you want to start and stop the IMS Apply program associated with running the Selector.

Scenario 5

You require ad hoc information.

Use IMS DPROP to keep duplicate copies of data, either constantly synchronous or synchronous to points in time, and to prevent inconsistencies between duplicate data copies. You can also use IMS DPROP to keep history data or audit data. You move and translate data but are not concerned about keeping data in sync. You can periodically check data updates.

You check your data by assigning START TSMs from a timestamp you specify using the TSMF. Your timestamp does not prevent the database from being updated during the time specified.

The Selector might encounter IMS UOWs that cross the start time, indicated by no 'first' record being located. You can override the normal Selector protection provided against such invalid UOW by specifying INVUOW=IGNORE in the Selector control statements.

Remote site considerations

The following LOG-ASYNC propagation files are used at both the IMS source and DB2 target sites:

IMS DBDLIB

Used at the source site by IMS and the SCF Apply utility. Used at the target site by the IMS DPROP MVG for creating and altering propagation requests. Used by the CCU for checking and restoring data consistency.

Initial data extract file

Created at the source site by DataRefresher or another extract program. Used at the target site to load the DB2 tables.

PRDSs

Created at the source site by the Selector and used at the target site by the Receiver to update the DB2 tables.

Group Definitions file

Created at the target site by the Group Unload utility from the propagation request definitions and used at the source site by the SCF Compare and Apply utilities to update the SCF.

IMS HD unload file

Created at the source site by the IMS HD Unload utility and used at the target site by the CCU to check the consistency of the IMS and DB2 data.

If the IMS and DB2 systems are on:

- The same MVS image, or if the files can be defined on shared DASD volumes, no file transport mechanism is required
- Remote MVS images, you must provide a mechanism to transport the files between the source and target image.

The following sub-sections describe the considerations that apply to each type of file.

IMS DBDLIB

The IMS DBDLIB file must be transmitted from the source to the target site only when a change has been made to it. Change is infrequent, therefore you do not need to automate or link the transfer to any other jobs. You can use a simple transport mechanism such as the TSO Transmit/Receive commands.

Initial data extract file

The initial data extract file must be transmitted from the source to the target site only when a full extract and load is performed. Extract and load is infrequent; therefore, you do not need to automate or link the extract and load to any other jobs.

You can use DataRefresher for extract and transfer of the file to the target site.

PRDSs

Automate the processing of PRDS as much as possible so that you can run the Selector (which creates the PRDSs) and the PRU and Receiver (which processes PRDSs) many times during a day, as needed. Automation also ensures that the Receiver processes PRDSs in the same order as the Selector created them.

Automate the PRDS process by running a job at the source site that carries out the required steps and proceeds to the next step only if the previous step completes successfully. Steps for processing PRDSs are:

1. Run the Selector to create the PRDS at the source site.
2. Send the PRDS from the source to the target site.
3. Receive the PRDSs at the target site.
4. Submit a job at the target site to:
   a. Run the PRU to register the PRDS
   b. Run the Receiver to process the PRDS

If the IMS and DB2 systems are on different MVS images and the PRDSs are not created on shared DASD, you must transport the PRDSs from the IMS image to the DB2 image before they can be registered and applied to the target DB2 tables by the Receiver. In order to automate the process, you need a transport mechanism that completes successfully only when the file being transported is received at the target site. Without such a mechanism, you would have to submit the job to run the PRU and the Receiver manually after verifying that the PRDS had been received at the target site.

To automate PRDS processing you can use:

- The IBM NetView File Transfer Program to transport files. Refer to the documentation library for NetView File Transfer Program for MVS for further information.
- A program that carries out the required steps and proceeds to the next step only if the previous step completes successfully. Steps required are:
  1. Convert a PRDS to FB80 format.
  2. Append the FB80 PRDS to the end of a piece of JCL.
  3. Submit the JCL at the target site. The JCL runs a program at the target site that:
     a. Convert the PRDS back to VB format
     b. Run the PRU

c. Run the Receiver

If you want to run the Selector several times before transporting the PRDSs and applying them, you could set up the following jobs:

- Job 1 runs the Selector to produce a PRDS for each selected propagation group.
- Job 2 runs the Selector to produce a PRDS for each selected propagation group. Job 2 also transports all PRDSs produced by either Job 1 or Job 2 to the target site, receives the PRDSs, and submits Job 3 to register the PRDSs, and run the Receivers.

You could run Job 1 four or five times throughout the day and run Job 2 during end-of-day processing.

Notes:

1. The data set names of all PRDSs created by the Selector are stored in the PRDS Register File. You can use the file to provide the list of PRDSs that need to be transported to a remote site.
2. Or, since the Selector creates a defined number of PRDSs on each run (one PRDS for each selected propagation group) and the data set name of each must be hard coded in the Selector JCL, you could also hard code data set names in the transport step transportation. Because the Selector JCL references PRDSs by their GDG relative index number, reference them in the transport step the same way.

Since the Selector can create a number of PRDSs that can be transported to different sites and processed by different Receivers, you might have to repeat some or all of the PRDS processing steps. For example:

- If you must transport a PRDS to more than one remote site, you must receive, register, and apply the PRDSs at each target site.
- If you must transport more than one PRDS to a single remote site, you *must* use different Receivers to process each PRDS, so that each Receiver runs separately.
- If you run the Selector more than once before transporting and applying PRDSs, you must transport, receive, register, and apply each PRDS at the target site.

If you implement automated PRDS processing in a remote site environment, use a data set naming convention for the PRDSs received on the target site that allows you to run the PRU with the same control statements.

Group Definitions file

A Group Definitions file must be transported from the target site to the source site each time a change is made to a propagation request contained in a propagation group and each time the assignment of propagation requests to Receivers changes. Change is infrequent and can be handled manually.

IMS HD unload file

An IMS HD unload file is required at the target site only when you need to check consistency between the IMS and DB2 data. The consistency check can be handled manually.

Recommendations on LOG-ASYNC database administration

IMS DPROP provides data integrity when moving and applying data from source to target. IMS DPROP protects against duplication of updates and missing updates within one DB2 system. Adhere to recommendations presented in this section to ensure protection of your data.

A propagation group is sourced by a set of segments. The set of segments should relate to one another in an application. If an application updates two segments, and the segments are split between different propagation groups, the source UOW is also split. UOW consistency cannot be maintained in the DB2 apply.

Generally, it is recommended that you define one Receiver for each PRDS propagation group so that all propagation requests for the set of segments in the propagation group are assigned to the one Receiver. However, you might have reason to define more than one Receiver to a propagation group.

When you use multiple Receivers (connected to the same DB2) for one propagation group, group the propagation requests for a Receiver; for example, group propagation requests for related tables under one Receiver. If a Receiver processes a UOW, and not all of the propagation requests applicable to the UOW are assigned to the Receiver, then commit consistency of the source UOW cannot be preserved across the DB2 tables to be updated.

If propagation requests for related tables are split between multiple Receivers, you might encounter propagation errors. For example, assume Table A is a 'parent' table to Table B, through referential integrity rule ON DELETE RESTRICT, and the propagation requests for Table A and B are split to two different Receivers. If the Receiver which updates Table A is run first, and a source UOW originally contained deletes for B and then A, you could encounter an unnecessary propagation failure when an attempt is made to delete Table A first.

If no RIRs are in effect, then you can achieve commit consistency even though propagation requests are divided among multiple Receivers. However, if one of the Receivers ends abnormally, then you might lose commit consistency between the propagation requests divided among multiple Receivers. If using PATH=DENORM, you should run propagation requests with the propagation requests of the parents and ancestors providing the path data, to reduce the risks associated with PATH=DENORM.

Although defining propagation groups and executing multiple receivers adds greater flexibility and improves the performance of the product, you are responsible for creating an IMS DPROP strategy that does not compromise the level of consistency desired.

Setting synchronization points

One of the greatest challenges you face as the administrator of LOG-ASYNC propagation is the management of changes and recovery that affect the source or target data for propagation. Just as you need initial synchronization when first implementing LOG-ASYNC propagation, you need synchronization when recovering from failures and making changes to mapped data.

This section introduces methods of achieving synchronization. In both methods, if outstanding PRDS exist at the time that you are planning to synchronize, you must

receive or drop the PRDS before synchronization. The following subsections present two methods and some considerations for when you are synchronizing your system.

Method 1

You can use the timestamp marker facility to create a quiesce TSM for a database whenever the database is in read-only status. At least one of the timestamps must be created and assigned to a 0302 (propagation group/database) record for use as a valid selection start time. This method assumes that IMS DBDs are coded correctly

You can create additional quiesce TSMs at each point when the database is in read-only status. The TSMs are stored as a 0202 record, but not assigned to a 0302 record as a start time. The TSMs represent potential valid start times for the Selector, that you can use as part of your recovery or change plan.

IMS recovery requires an image copy of the database. Timestamp recovery of the database requires recovery points after the image copy when the database is quiesced. At points in time when the database is quiesced for IMS recovery planning, you can create quiesce timestamps for propagation recovery planning. For greater efficiency and effectiveness, you might want to merge you plan for propagation recovery with your IMS recovery plan. It is also recommended that you make a copy of the SCF and the ULR data set.

The following example illustrates the use and advantage of keeping the quiesce timestamps:

Example: Keeping the quiesce timestamp

You find that the mapping information that was provided to build a propagation request is incorrect (and EXIT parameters on the IMS DBD are not part of the problem). By the time you discover the problem, much data has already been propagated. Furthermore, it would be difficult to correct the problem immediately. You'd rather set ERROPT IGNORE, and to accept the few inconsistencies that result between the IMS and DB2 copies of the data until you have time to make the repair.

To repair the situation, you need to correct the mapping information and regenerate the propagation request. You also need to repair the inconsistencies. You must create a new synchronization point between the IMS and DB2 copies of the data.

If you created quiesce timestamps for the IMS database, you can use a quiesce timestamp to create a synchronization point. How you proceed depends on what was created on the IMS side when the database was in READONLY status:

- If you created an image copy of the database at that quiesce point, you can use the image copy to create a 'shadow' copy of the IMS database as it existed at the quiesce point.
- If you only quiesced the database, you can perform a timestamp recovery to create a 'shadow' copy of the IMS database.

You cannot use a fuzzy image copy for the shadow copy, but when the database is quiesced, you can create a shadow copy by doing a timestamp recovery using the fuzzy image copy plus the logs up to the quiesce point.

When you use the IMS database recovery utility to create a shadow copy, you can use DBRC to generate the JCL, but you must perform the actual recovery process

outside the scope of DBRC control. Also be careful to ensure that the recovered data set name (the shadow copy), has a different data set name from the production database.

You can use the shadow copy as input to the CCU, and synchronize the DB2 copy to the shadow copy of the IMS database. Alternatively, you can use the shadow copy as input to a full extract to create a consistent copy. When the copies are resynchronized, based on the corrected mapping, you have a new synchronization point. The quiesce timestamp is now assigned to the 0302 record for the mapping, and selection can start from this point in time.

You can use the CCU or a full extract in the resynchronization process. Using the CCU can be faster than performing a full extract. However, the CCU does have limitations for PRTYPE=F and L and user mapping cases.

Quiesce TSMs are stored in 0202 records, and apply to the *database*. They can be assigned to a particular *propagation group*, and then are stored in the 0302 record for the propagation group. Assigning a new start time for a database in one propagation group does not affect another propagation group using the same database. However, the entire database within the propagation group is affected by the new start time. All propagation requests for which the propagation group or database is source must be resynchronized prior to using the new Selector start time.

In this synchronization process, the actual IMS source database does not need to be quiesced. The process is using an old quiesce point and a shadow copy. Sites that need continuous availability value this type of recovery. Quiesce points are set when it is convenient, as opposed to on a 'demand' basis due to the need to resynchronize.

Method 2

In Method 1, it is assumed that the IMS DBDs are coded correctly. Method 2 assumes that the IMS DBD has not been coded correctly. Perhaps NOCASCADE is specified when it should be CASCADE, resulting in deletes not being propagated.

Because DBDs are incorrect, the IMS data which has been collected is not correct. If you return to a previous quiesce TSM, and start selection, you could recreate the same problem. Instead, you need a quiesce of the IMS database. You need a quiesce in order to make the changes to the IMS DBD. You then create a new quiesce TSM while the database is quiesced, assign this timestamp to the 0302 record, and start selection from the new start time.

You have flexibility using this synchronization method to minimize the duration of outage. You can choose to not resynchronize or re-extract from the actual database. You can restart the database as soon as the repairs are made to the DBD. You can perform the resynchronization from a shadow database, created from the newly created quiesce point. If the outage time is not a concern, you can make the extracts or CCU executions from the actual source database while it is in read-only status.

Additional synchronization considerations

In normal circumstances, the Selector only selects data once, and the Receiver applies the data only once for any propagation request. During resynchronization, the Selector might re-select data and the Receiver might re-apply data.

During recovery, you might reassign a propagation request or set of propagation requests to another Receiver. You would then use the recovery Receiver to recover the propagation requests back to the point where they can be merged with the original Receiver. During the recover process, both Receivers have the same PRDS position information. If you want to merge the propagation requests back with the original Receiver without resynchronizing the data, the propagation requests can be reassigned using the MERGE=FORCE option.

You can use DBRC to create the JCL that creates the IMS shadow database. However, you must perform the actual recovery or restore outside of the scope of DBRC.

Changing propagation requests or propagation groups

This section describes possible actions that you must take if you change, add, delete or move propagation requests or propagation groups.

The IMS DPROP directory contains the definitions of propagation requests and propagation groups at the target site. The Selector control file (SCF) records the IMS source of the propagation groups at the source site.

Changes you make to propagation requests or propagation groups are made at the target site but must be reflected in the SCF at the source site when:

- You change the IMS source for a propagation request.
- You add a propagation request to a propagation group, using the SCU ASSIGNPR control statement to assign the propagation request to a Receiver
- You remove a propagation request from a propagation group, using the SCU DELETEPR control statement to remove the propagation request from a Receiver
- You move a propagation request one propagation group to another, using the SCU ASSIGNPR control statement to reassign the propagation request from one Receiver to another Receiver

You use the Group Unload utility, the SCF Compare utility and the SCF Apply utility to ensure changes made at the target site are reflected at the source site.

You might also want to refer to “Replacing a propagation request” on page 115.

Synchronizing propagation request and propagation group changes

When changes you make to a propagation request or propagation group result in a new IMS database or segment becoming the source for a propagation group, then you must synchronize IMS and DB2 databases by creating a synchronization point for the database and the mapped target tables, using either of the methods described in “Setting synchronization points” on page 192. Because TSMs are set at the database level, the process is not complex.

Adding a segment to the source of a propagation group or changing any field information makes the synchronization process more complex. Adding a segment requires that you coordinate other segments already subject to propagation.

For example, assume that a propagation request is being changed from mapping case 1 to mapping case 2. The target table needs data from an extension segment X of the entity segment E currently mapped. You must:

1. Change the DBD for segment X to add the EXIT= keyword.

2. Alter the target table.
3. Alter the propagation request.
4. Run the GUU and SCF Compare and Apply utilities.
5. Set the IMS database to read-only status.
6. Perform an extract for the altered table.
7. Set a new start point for subsequent selection of the data for this propagation request.

You must establish a synchronization point for the whole database within the affected propagation group or propagation groups. When you do not need to change the IMS DBD, you can use either synchronization method. If you must change the DBD, you must use the method that involves creating a new quiesce timestamp as explained in “Setting synchronization points” on page 192.

To add a new segment to an existing propagation group:

1. Receive any outstanding PRDSs before applying the synchronization procedures.
2. Alter the IMS DBD, if needed, and perform the necessary DBDGEN and ACBGEN. Transport the DBDLIB to the target site, if necessary.
3. Alter or create the DB2 target table.
4. Alter or create the propagation request.
5. Perform synchronization for all mappings in the affected database for the affected propagation groups, using the appropriate synchronization method. The availability of the IMS source database depends on the method used for synchronization.
6. Make the changes in the Selector control file by running the GUU and the SCF Compare and Apply utilities.

Error recovery

This section describes changes you might need to make if you encounter database problems, mapping failures, or other failures resulting in inconsistencies between duplicate database copies. Topics included in this section are:

- IMS database recovery
- DB2 table recovery
- Failures because of incorrect mapping
- Failures because of corrupt PRDSs
- Failures because of corrupt CDCDSs or SLDSs

IMS database recovery

Failures that involve an IMS database might require a full or partial, also known as timestamp, recovery. No facility exists for coordinated full recovery between IMS and DB2 databases. Instead, you must perform a partial recovery, which is a recovery of an IMS database, and take special actions to synchronize target DB2 tables mapped from the failed database.

Performing a recovery of an IMS database should have no effect on the asynchronous propagation of the database. No updates should be lost. However, you might want to run the Selector and Receiver using the recovery quiesce point as the STOP time. After the IMS recovery, you can run the CCU to check the subsequent consistency of the databases.

During a timestamp recovery, IMS updates might exist on the log for the time period past the time for which the database is being recovered. Additionally, these updates might already have been applied to the DB2 target tables. This problem could occur when a new application is brought into a production environment causing the databases to lose synchronization.

You must quiesce the IMS database to perform recovery. You must establish a synchronization point for the propagation requests affected by the propagation group and database. Because there might be data on the IMS log which should not be selected or propagated, your synchronization method should use the new quiesce point for the database recovery.

DB2 table recovery

DB2 failures might require recovery. A full DB2 recovery should not affect propagation. The Receiver can wait and hold updates until the table is fully recovered.

A partial DB2 recovery is more of a problem. The table is recovered to a previous point in time, and could be missing updates that have been applied to the IMS source database. PRDSs that have new updates for this table might exist. The PRDSs updates might fail during the Receiver run because the table is out of synch. Additionally, newer updates are in the IMS logs or stored as ULRs in the control database. You must ensure these updates are applied to your DB2 table.

You must synchronize at the table level, which is also the propagation request level. You must also provide a new selection start point after establishing synchronization. Because start timestamps are set by IMS DPROP at the IMS database level, you need a synchronization point for the IMS database that sources the propagation request. You can use either of the two synchronization methods described in “Setting synchronization points” on page 192 can be used to establish the synchronization point.

Failures because of incorrect mapping

The MVG attempts to completely verify the mapping definitions, but you could encounter errors during data translation or updates to DB2 after propagation has been started for a propagation request. For example, IMS performs no checking of the type of data stored in a segment. Therefore, at run time, an IMS application can insert data in an IMS segment data that does not match the description of the data supplied in the IMS DPROP mapping definition.

In synchronous propagation, the error is noted in the same UOW as the IMS insert being performed, and can be rejected, forcing an IMS backout of the bad update. In asynchronous propagation, the IMS change is logged, and the problem with the data is not found until the data is to be applied at the target. At this point, the update process notes the failure. If the error option is IGNORE, the failure is noted but processing continues. If the error option is BACKOUT, then the Receiver backs out the current DB2 UOW and terminates.

The recovery process for failures caused by mapping problems varies depending on whether the mapping itself is incorrect, or the source data is incorrect, and on whether the update failures were ignored or not.

If the mapping definition is incorrect, and no updates were ignored (ERROPT=BACKOUT), your recovery process can be simple. Alter the propagation request to reflect the correct mapping, and run the Receiver again. The Receiver

starts from the point of failure. During test and early implementation stages of asynchronous propagation, ERROPT=BACKOUT is recommended because of the ease of recovery.

If the DB2 tables are affected by the bad mapping, with updates being made for the DB2 table, then your recovery process is more complex. Or, if you determine that the mapping definition is incorrect, and updates have been ignored (ERROPT=IGNORE), your recovery process is complex. In both these instances, you must correct the propagation request and also recapture the lost updates.

You must create a synchronization point to repair the inconsistencies and allow selection to resume. Because mapping is the problem, not the IMS data, you can use either of the two synchronization methods described in “Setting synchronization points” on page 192 to establish the synchronization point.

If your mapping problem includes an error in the IMS DBD exit definition, then you must establish the synchronization point using the new quiesce point of the database corresponding to when the DBD changes were made.

Failures because of corrupt PRDSs

If one of your PRDSs is corrupt, run the Selector again using the same:
- Database start times and propagation group stop times
- SCF and ULR data set

As described in “Establishing the start conditions for LOG-ASYNC propagation” on page 100, it is strongly recommended that you make copies of the SCF and the ULR data set on a regular basis. Then, if you need to recover from a corrupt PRDS, you can restore the copies of the SCF and the ULR data set before running the Selector again.

Failures because of corrupt CDCDSs or SLDSs

If the Selector encounters an error when it tries to read a CDCDS, delete the CDCDS and re-run the Selector, which will use the corresponding SLDS.

If the Selector encounters an error when it tries to read a SLDS, ask your IMS DBA to make the SLDS specified in the SECLOG record available as if the record were in the PRILOG record by:

1. Deleting or renaming the SLDS whose DSN is in the PRILOG record
2. Renaming the SLDS whose DSN is in the SECLOG record to the DSN in the PRILOG record

Timestamp Marker Facility (TSMF)

Invoke the timestamp marker facility by using the SCU. The timestamp marker facility provides a simple and reliable method for creating and deleting timestamps used by the Selector. The Selector uses timestamps to delimit the time periods for which IMS updates are selected for propagation to DB2.

Refer to Chapter 10, “Performing LOG-ASYNC propagation,” on page 149 for details of how the Selector uses the timestamps. Refer to the SCU description in the *IMS DataPropagator for z/OS Reference* for details of how to create and delete timestamps. Topics included in this section are:
- Types of times and timestamps
- Where timestamps are stored

- How timestamps are displayed
- Start time granularity
- Stop time granularity

Types of times and timestamps

The Selector needs timestamps to delimit the time period during which updates are selected from the IMS log. IMS DPROP parameters for timestamps are:

Selector Start time
Determines the time the Selector starts searching IMS logs for data to be propagated.

Selector Stop time
Determines the time the Selector stops searching IMS logs for data to be propagated.

Database Quiesce timestamp
Sets the time in which an IMS database is in a quiesced, or “read-only” state.

Group Database Start timestamp
Sets the time at which propagation for a database within a propagation group is to start.

Group Stop timestamp
Sets the time at which propagation for a propagation group is to stop.

Where timestamps are stored

All timestamps are stored in the Selector control file:

Database Quiesce Timestamps
Stored in the 0202 Database Quiesce Timestamp record.

Group Database Start Timestamps
Stored in the 0302 Group Database Start Timestamp record.

Group Stop Timestamps
Stored in the 0305 Group Stop Timestamp record.

Refer to the *Reference* for details on the Selector control file records.

How timestamps are displayed

The SCF Administration control statements include LIST control statements, which allow you to list all information in the Selector control file for a specified database or group.

Timestamp information stored in the 0202, 0302 and 0305 records is included in the display.

Related Reading: See the*IMS DataPropagator for z/OS Reference* for details of the Selector control file control statements.

Start time granularity

Start time granularity is at the database level within a propagation group. For a database in a propagation group, you can specify a timestamp to indicate where selection should begin for the database. You must assign an initial start timestamp before you begin to propagate the database. You can also assign a timestamp for a recovery, re-synchronization, or re-extract.

Stop time granularity

Stop time granularity is at the propagation group level. You can use the TSMF to create a propagation group stop timestamp for each propagation group. If you do not create a stop timestamp for a propagation group, the Selector creates one, based on control statement input or the propagation log records.

Selector time zones

The Selector is designed to run in MVS TOD time, which is consistent with the timestamps that are used for IMS logs. However, when you use the SELECT control statement to specify a timestamp, you use local time (wristwatch time) in ISO/DB2 format. The timestamps in ISO/DB2 format you put into the Selector are adjusted by the value in SYS1.PARMLIB and converted to MVS TOD format. Both the local ISO/DB2 format time value and the adjusted MVS TOD format time value are stored in 0302 (propagation group/database) records and 0305 (propagation group/stop time) records in the SCF.

When the Selector is initialized and uses a stop time from a timestamp, it:

1. Reads the MVS TOD value from the 0302 records for each selected propagation group that contains a database that has specified a new start time.
2. Calculates the local time based on the value in SYS1.PARMLIB
3. Compares the calculated local time with the local time stored in the 0302 record. The Selector then issues an error message and terminates processing.

Log selection considerations

Because of differences in precision between:

- DBRC log start and end timestamps
- IMS log record TOD timestamps
- DB2/ISO format timestamps

logs that immediately precede or follow the Selector time span might be required for processing. A TOD or DB2/ISO timestamp which, when truncated to DBRC precision, equals the end time of a log (and the start time of the next log in sequence), and makes it difficult to discern which of the logs contains the relevant log records without reading both logs that immediately precede and follow the Selector time span.

For example:

- The earliest of all propagation group start times, converted to DBRC precision (1/10 second), equals the end time of a log.

  The log should not be required; however it might contain records whose timestamps are later than the Selector start time, for example, when the stop type for the propagation group in the previous Selector run was not INTERIM.
- The latest of all propagation group times, converted to DBRC precision (1/10 second) equals the start time of a log.

  The log should not be required; however it might contain records whose timestamps are earlier than the Selector stop time, for example, when the stop type for the propagation group is not INTERIM.

RUP control statement: TRACE

TRACE is the only RUP control statement available for LOG-ASYNC propagation. Use TRACE to track activities of jobs that call RUP for LOG-ASYNC propagation. You can put multiple TRACE control statements in a single //EKYIN DD statement in your IMS batch or dependent region JCL.

Use TRDEST to direct the output of a trace to a particular destination. In job steps used to call RUP for LOG-ASYNC propagation and for IMS DPROP utilities, use TRDEST to direct trace output to:

- //EKYLOG DD statement
- //EKYTRACE DD statement

Trace output from //EKYTRACE is formatted.

Depending on the level of trace selected, you can receive a large volume of trace data. You should be aware of the impact tracing can have on the IMS log, or allocate an adequate amount of space for the //EKYTRACE or //EKYLOG data set.

Related Reading: For syntax diagrams, options, and detailed information on these control statements, refer to the *IMS DataPropagator for z/OS Reference*.

Chapter 14. Verifying Data Consistency (CCU)

This chapter is an overview of the IMS DPROP Consistency Check utility (CCU), which checks the consistency of propagated data and generates repair statements if errors are found.

This chapter also describes the phases and stages associated with running the CCU. More detailed information on the CCU is in the *Reference*, including sample JCL.

Overview of the CCU

Use the CCU to check consistency between the IMS and DB2 copy of the data.

The CCU:

- Supports only propagation request for generalized mapping cases
- Does not support user mapping (PRTYPE=U)
- Uses the IMS DPROP directory to determine the relationship between IMS segments and DB2 tables
- Uses RUP and HUP (for synchronous) to follow mapping done during propagation
- Compares IMS segments and related DB2 rows for data existence and compares IMS fields and corresponding DB2 columns for data content
- When inconsistencies are detected, generates repair statements for propagated IMS segments (synchronous only) and DB2 tables that are in error.

The CCU runs as a relational application in an IMS batch job step (DBBBATCH or DLIBATCH), or IMS batch message processing region (BMP, IMSBATCH).

The CCU run and repair process is illustrated in Figure 48 on page 204. If the CCU finds inconsistencies, it generates an SQL call statement to correct them.

If SQL statements are generated, you can run them through DSNTEP2 or DSNTIAD to restore DB2 tables to the same data content as the IMS database. For synchronous propagation, if DL/I statements are generated, you can run them through DFSDDLT0 to restore the IMS database to the same data content as the DB2 tables.

CCU Execution and Repair Process

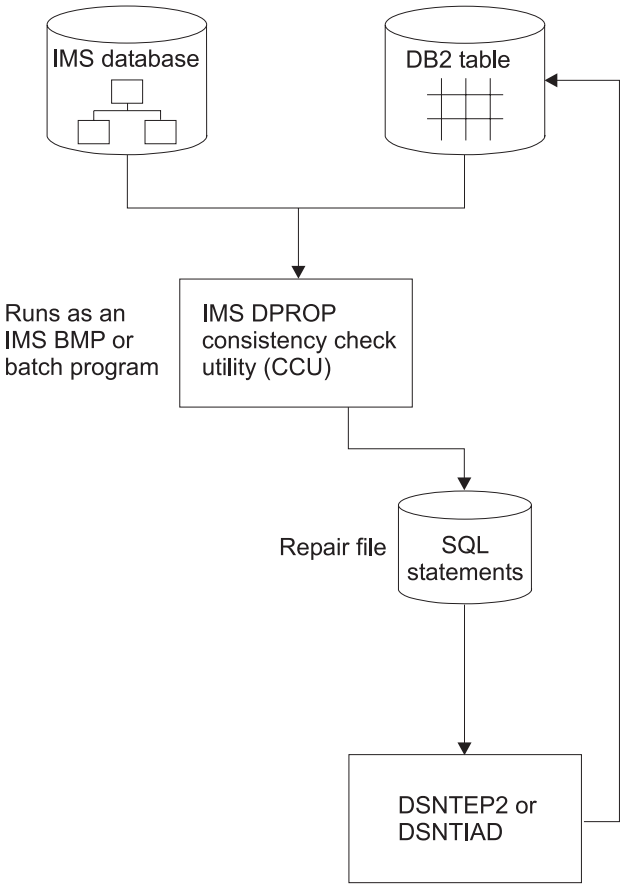


Figure 48. Process for running and repairing the CCU

The following sections discuss:

- When to use the CCU
- Considerations for LOG-ASYNC propagation, synchronous propagation, and user asynchronous propagation
- Considerations when concurrent updates are being done
- Data availability when the CCU is active
- DB2 referential integrity constraints

When to use the CCU

You can use the CCU:

- Periodically, to verify the consistency of your propagated data
- As a verification tool for testing propagation requests that you are developing
- After implementing new or changed propagation definitions
- Following a database reload in either IMS or DB2
- After propagation has failed
- After synchronous propagation has been suspended, deactivated, or emergency stopped
- After running database repair programs with PROP OFF control statements

- After encountering operator or application errors, for example, user exit

You might also find other reasons to use the CCU at your installation.

CCU considerations for LOG-ASYNC propagation

In LOG-ASYNC propagation, unlike synchronous, there are fewer times when tables and IMS databases will be consistent. To check the consistency between the IMS and DB2 data, the copies must be at a logical point of consistency which might not occur without your intervention. If you want to make consistency checks, you must create logical synchronization points.

To create a logical synchronization point, you need a database quiesce point to IMS data copy at a point in time. To bring your DB2 copy to the same point as the IMS copy, run the Selector using the quiesce point as the stop time, and have run the Receiver to apply the resulting PRDS. If you keep the IMS database in read-only state throughout the select and receive process, then you can run the CCU against the actual IMS database and the DB2 copy.

Or, you can make the IMS database available for update immediately after creating the quiesce point and creating a shadow copy of the IMS database. The shadow copy represents the state of the database at the quiesce point and can be used as input to the CCU.

If you run the CCU using a shadow copy on a regular basis, you can plan for recovery in yet another way. At the successful conclusion of the CCU run, you can create a DB2 quiesce point for the target table and use the DB2 quiesce point as input to the DB2 recovery process. You then have matching recovery points in IMS and DB2.

Only when IMS and DB2 are on the same MVS image can the CCU read both IMS and DB2 copies of the data. You must add additional function to the CCU in order to support other configurations. For example, as a substitute for reading the IMS database, the CCU can accept an HD Unload MVS file as input, on the MVS image where DB2 and the base IMS DPROP product reside. The IMS HD Unload utility supports all propagation database types except for DEDBs. When the IMS database is quiesced, run the HD Unload utility against the database. Then make the database available for update. After the DB2 copy is brought up to date with the IMS copy, use the HD Unload file as input to the CCU. Only the hashing method of the CCU is supported for the processing of HD unload type input.

For the synchronization method involving an:

- Old quiesce point, you must use an unload file or shadow copy of IMS as input to the CCU
- New quiesce point, you can use either the actual IMS database, a shadow copy, or an unload file.

There is a trade-off between choosing the simplest method of running the CCU, and the length of time that the IMS database is unavailable for update:

- The simplest CCU method is to run the CCU against the actual IMS database while it is not being updated. In this case the IMS database is unavailable for update for the time it takes to run the CCU.
- The next simplest method is to use the HD unload file created from the IMS database. In this case the IMS database is unavailable for update for the time it takes to create an unload file.

- The least simple is to create an IMS shadow database. In this case the IMS database is unavailable for update for the time it takes to create a recovery point.

Local MVS image

You can only run the CCU with an IMS HD unload file. You must make the unload file available to the target MVS image where you run the CCU, for example, by creating it on shared DASD.

Remote MVS image

You can only run the CCU with an IMS HD unload file. You must make the unload file available to the CCU by transporting the file to the target MVS image where you run the CCU.

CCU considerations for user asynchronous propagation

When you are performing user asynchronous propagation, you should run the CCU after updates collected for DB2 are applied. Otherwise, IMS databases might reflect updates not yet made to DB2 tables. With user asynchronous propagation, out-of-synchronization conditions are normal, and data is synchronized for only brief periods. The CCU reports inconsistencies until updates are made to the DB2 side.

If the user asynchronous propagation file containing DB2 updates is destroyed or damaged, running the CCU might be one way to recover the changes intended for DB2.

If IMS and DB2 are on different MVS images, you can provide the CCU with an IMS HD unload file as a replacement for the IMS database. If you do, the CCU compares the IMS data in the HD unload file with the data in the DB2 tables.

Considerations when concurrent updates are being performed

Avoid running the CCU concurrently with updating applications because some changed data might be flagged as a data mismatch. The CCU detects and eliminates many of these pseudo-errors in a later phase of its processing. Situations that cannot be eliminated are reported to you by the CCU for verification. Situations reported to you are when PRTYPE=L or F and the CCU finds a DB2 row without a corresponding IMS segment (for IMS-to-DB2 propagation, MAPDIR=HR). When analyzing CCU reports, you must then determine which of the reported inconsistencies are real errors.

Concurrent updating might also result in a longer elapsed time for execution of the CCU. See “Running the CCU” on page 207 and the *IMS DataPropagator for z/OS Reference* for more detailed information.

Data availability

When the CCU is accessing IMS data, databases can be in either update or read-only mode. The CCU generates a PSB with a default PCB processing option (PROCOPT) of G.

DB2 tables can be in read-write or read-only mode. The application plan for the CCU should be bound with cursor stability (CS).

DB2 referential integrity constraints

You can use the SQL repair file, created by the CCU when inconsistencies are found, to rebuild consistent data in the DB2 tables.

If RIRs are defined for the DB2 tables needing repair, be aware of the sequence in which you apply the repair statements. For example, if inserts or updates are made to child tables before corresponding repairs to parent tables, the referential integrity enforced by DB2 might cause some repair statements to be rejected. Or, if deletes are applied to parents before children, the deletes might fail with ON DELETE RESTRICT.

You should process repair statements in a sequence that is appropriate for your RIRs.

If you have not implemented DB2 RIRs, make sure the repair statements that you choose to apply do not cause logical inconsistencies within the DB2 tables.

Running the CCU

This section discusses the phases and stages associated with running the CCU. For more information on CCU control statements and JCL, refer to the *IMS DataPropagator for z/OS Reference*. The following sections provide:

- A summary of the phases of the CCU
- CCU verification techniques
- Types of inconsistencies and generated repair statements
- Suggestions for ways to reduce large numbers of inconsistencies
- Some reasons for inconsistencies

Phases of the CCU

The phases of CCU processing are:

Initialization
Checks your input control statements, collects IMS DPROP directory information, and builds the mapping control blocks needed by the CCU.

Read and compare
Reads and compares both IMS databases and DB2 tables. The CCU completes processing if no inconsistencies are found. Depending on which CCU verification technique you use, the read and compare is done in three phases (hashing technique) or one phase (direct technique). See “CCU verification techniques” on page 207

Error location
Relocates any mismatches or inconsistencies that are found. If there are concurrent updates and PRTYPE=E, then the CCU finds and eliminates from the set of CCU mismatches many of the data mismatches caused by the concurrent updates. The CCU writes the remaining inconsistencies to a report data set and creates the error repair files containing the SQL and DL/I call statements needed to reestablish consistency.

You can run all CCU phases in a single multistep job, or in individual jobs. Running the CCU in individual jobs might give you some benefits in parallel and independent processing during the read phase. Practices and procedures at your installation should determine how you run the CCU.

CCU verification techniques

The CCU uses two verification techniques, direct and hashing. You control which technique the CCU uses by the keywords you specify on CCU control statements.

Direct technique

Use the direct technique when IMS segments can be retrieved and checked in the same key sequence as related DB2 rows. You must define both the IMS databases and the DB2 tables on the same MVS system. The direct technique does not support an HD unload file as input for the IMS data.

Using the direct technique, each IMS segment to be verified is compared to its corresponding DB2 row. Any mismatches are detected and passed to the error location phase.

If few inconsistencies exist, the direct technique might require the least amount of elapsed time.

Hashing technique

With the hashing technique, the read and compare is done in these three phases:

- IMS read
- DB2 read
- Compare

During the read phases, various internal and external totals are created from reading the IMS databases and related DB2 tables. During the compare phase, the totals are compared and used to determine if inconsistencies exist.

You can use the hashing technique for all IMS database organizations supported by IMS DPROP.

Relating to IMS key retrieval sequences, you must use the hashing technique:

- When you cannot retrieve IMS segments in ascending key sequence, as with HDAM and DEDBs
- If retrieval by IMS key sequence does not match DB2 key sequence.

If you try to use the direct technique, a number of pseudo-mismatches are created and passed to the CCU error location phase, significantly impacting the elapsed time and usability of the CCU.

Types of inconsistencies and generated repair statements

When the CCU finds an inconsistency between IMS and DB2 data, the inconsistency is put in one of three categories:

1 An IMS segment is found, but the DB2 table row does not exist.

2 A DB2 table row is found, but the IMS segment does not exist.

3 Both the IMS segment and the DB2 table row are present, but the data content, excluding the IMS and DB2 key fields, is not the same.

If the CCU finds any data inconsistencies, it creates repair statements based on the mapping direction:

- For one-way IMS-to-DB2 propagation, the CCU generates SQL statements to update the DB2 copy and make it consistent with the IMS copy.
- For one-way DB2-to-IMS synchronous propagation, the CCU generates DL/I call statements to update the IMS copy and make it consistent with the DB2 copy.
- For two-way synchronous propagation, the CCU generates SQL statements to update the DB2 copy and make it consistent with the IMS copy. The CCU also generates DL/I call statements to update the IMS copy and make it consistent with the DB2 copy.

If the inconsistency is in category

1 The CCU generates an SQL INSERT statement for the missing DB2 row, or a corresponding DL/I DLET call for the IMS segment, or both.

2 The CCU generates an SQL DELETE statement, or a corresponding DL/I ISRT call for the missing IMS segment, or both.

3 The CCU generates an SQL UPDATE statement for the inconsistent columns, or a corresponding DL/I REPL call for the IMS segment, or both.

For every data inconsistency, you must determine whether you want to make data consistent by applying either:

- The generated SQL statement to the DB2 tables using the DSNTEP2 or DSNTIAD programs of DB2
- The generated Repair/ DL/I call for the IMS segment using the IMS test program DFSDDLT0

The CCU writes the generated repair statements to sequential files that you can edit with TSO/ISPF before passing the files to DSNTEP2, DSNTIAD, DFSDDLT0, or any compatible program you provide.

Generated SQL repair statements

SQL statements are created with an asterisk (*) in the first position, so that DSNTEP2 or DSNTIAD treat the statements as comments if the file is accidentally used before proper preparation.

You must decide which portion of the CCU-generated file to use as input to DSNTEP2 or DSNTIAD. Before using the CCU-generated file, sort it with the DFSORT or an equivalent program and do other additional processing as described in the *IMS DataPropagator for z/OS Reference*.

Large numbers of inconsistencies

If an unusually large number of inconsistencies occurs, it might be more efficient to re-extract the IMS data and reload the DB2 tables, or to use the DLU, rather than reestablish consistency with CCU-generated repair statements.

To verify your propagation request, rerun the CCU after reloading the data.

Reasons for inconsistencies

Some reasons for inconsistencies between propagated IMS and DB2 data are:

- Propagated tables or databases are updated but not propagated. This occurs, for example, with one-way IMS-to-DB2 propagation if appropriate DB2 security definitions are not in place to prevent users from updating DB2 tables; or it can happen with DB2-to-IMS synchronous propagation if DB2 tables are updated in a non-IMS environment.
- Referential integrity constraints in DB2 do not match the IMS hierarchy when the update is made.
- Faulty database recovery in either IMS or DB2.
  - The recovery are not applied to both copies of the data
  - Erroneous logs or image copies are used in the recovery process, causing the recovered database to be inaccurate
- Either IMS or DB2 has an internal error
- Bad pointers in either an IMS database or DB2 table are encountered

- Mapping during extract and load is not identical to mapping during data propagation.
- Errors occur in the mapping logic of a user-provided IMS DPROP Segment or Field exit routine
- ERROPT=IGNORE is in effect, and errors are encountered
- LOG-ASYNC propagation, updates are not propagated before the CCU runs
- For user asynchronous propagation, updates are not propagated before the CCU runs.
- For DB2-to-IMS synchronous propagation, monitor class 6 are stopped.

There can be other causes for inconsistencies. Determine the cause and eliminate the problem if the inconsistencies are critical.

Chapter 15. Problem determination tools

This chapter explains how to get information about various database objects and system activities using IMS DPROP auditing and tracing facilities, message table, and CCU. These diagnostic tools can help you identify problems and resolve them.

You can also use IMS and DB2 when identifying and resolving propagation-related problems:
- IMS DFSERA10 utility, which reads and formats IMS log records
- DB2 DSNILOGP utility, which reads and formats DB2 log records

IMS DPROP trace facilities

Use the trace facilities to trace propagation activities of:
- RUP
- Selector
- Receiver
- IMS DPROP utilities

Start the IMS DPROP trace by putting a TRACE control statement in the //EKYIN file allocated to the job step in which IMS DPROP runs.

If you use a TRACE control statement in the //EKYIN file of a job step, then only the IMS DPROP activities of that particular job step are traced.

You can also limit tracing of propagation activities to or from specific DBDs and segments. When starting the IMS DPROP trace, specify on a DEBUG= keyword the type of information that you want to traced.

DEBUG level 1 produces in-core tracing written with low overhead into a wrap-around virtual storage trace table; the trace table is available in most storage dumps. Level 1 tracing is always active. Other debug levels write trace records either to a sequential output file or the IMS log.

You might want to look at the output written by DEBUG level 2. Level 2 output shows how IMS DPROP performed the data conversion, mapping, and propagation. During testing and diagnosis of IMS DPROP user exit activities, you might want to look at DEBUG level 4. The output of other levels is usually intended for IBM support personnel.

For other types of job steps, for example, IMS DPROP utilities or jobs calling RUP for asynchronous propagation, you can write trace output to the //EKYLOG or //EKYTRACE data set.

If you write the trace to the IMS log or to //EKYLOG, the trace is unformatted and, therefore, requires less CPU overhead, I/O, and external storage. You can also print and format trace records selectively. You can format the trace records with IMS DPROP EKYZ620X exit routine of the IMS File Select and Formatting Print utility (DFSERA10).

Related Reading: The *IMS DataPropagator for z/OS Diagnosis* contains detailed information on IMS DPROP trace functions.

Audit facilities

IMS DPROP records significant events in its audit trail. Information included in the audit trail are:

- Error messages issued by RUP and HUP
- CCU executions
- MVG executions
- Warning messages issued by the MVG
- DLU executions
- Selector executions
- Receiver executions

The following sections cover:

- Using SMF
- Audit extract utility and audit trail table
- Creating an audit trail
- Audit trail table security
- A comparison of audit and trace information
- How the audit trail works with CCU

Using SMF

IMS DPROP writes the records comprising its audit trail to SMF. IMS DPROP uses only one type of SMF record. When you customize IMS DPROP during installation, you define which SMF record type IMS DPROP uses. To start recording SMF records, update the SMFPRM*xx* member of SYS1.PARMLIB at your installation.

Related Reading: Additional information on this subject is available in the *IMS DataPropagator for z/OS Installation Guide*.

Audit Extract utility and Audit Trail table

IMS DPROP includes an Audit Extract utility (AUDU). AUDU extracts the IMS DPROP SMF records from a sequential file and loads them into a DB2 table. The AUDU sequential input file must be created by the SMF Dump Program (IFASMFDP) as described in *OS/390 MVS System Management Facilities*. When the records are loaded into a table, you can use QMF to query the audit trail information. You can also write your own SQL programs to extract information from the audit trail table.

The audit trail table is created during the IMS DPROP installation process. Contents of the audit trail table are the same as the audit trail SMF records.

Related Reading: The format of the audit trail table is described in the *IMS DataPropagator for z/OS Reference*.

The audit trail shows:

- When the CCU last ran
- When the data was last known to be consistent
- When the Selector last ran and the outcome
- When the Receiver last ran and the outcome
- When the RUP or HUP reported a problem
- If a propagation request regenerated

You might want to run the AUDU and load the audit trail table:

- Regularly, whenever SMF data sets are dumped using IFASMFDP.
- Whenever you must diagnose problems. The audit trail table usually contains a combination of historical archived SMF data, and actual current SMF data from the SYS1.MANx data sets.

You will need to set up a procedure for archiving audit trail table data.

Related Reading: Additional information, sample JCL, and control statements for AUDU and the audit trail table are described in*IMS DataPropagator for z/OS Reference*.

The Figure 49 summarizes the audit trail generation process.

IMS DPROP Audit Process

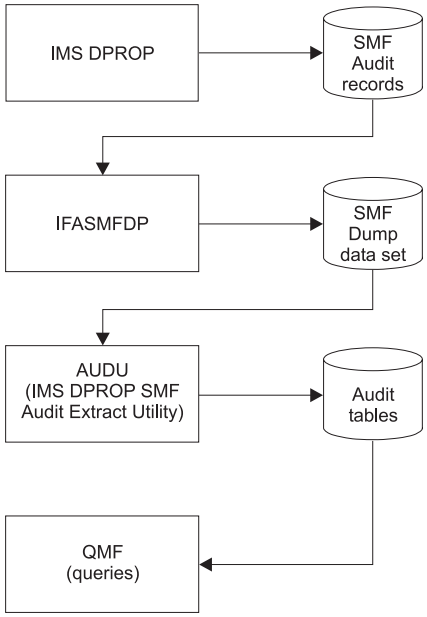


Figure 49. Overview of the IMS DPROP audit process

Creating an audit trail

The steps in creating an audit trail for IMS DPROP are:

1. Specify the SMF record code to be used by IMS DPROP.
2. Start recording SMF records by updating the SMFPRM*xx* member in SYS1.PARMLIB.
3. Create the audit trail table.
4. Bind a DB2 plan for the AUDU, and grant privileges to run this plan.
5. Grant SELECT privileges on the audit trail table.
6. Develop a procedure for running the AUDU to extract records from SMF and load them into the audit trail table.
7. Implement a maintenance procedure for the audit trail table.

Most of the steps involved in creating an audit trail are performed as part of installation (Step 1 to 5). Steps 3 on page 213 to 7 on page 213 do not apply to LOG-ASYNC Selector-only sites that do not have DB2 installed. See*IMS DataPropagator for z/OS Reference* for details of how to create an audit trail in a Selector site.

Audit trail table security

To access the information stored in the audit trail table, you must grant the SELECT privilege anyone who will use the information. People who SELECT, UPDATE, INSERT, DELETE, and maintain the table (for example, deleting old records) must be granted update privileges.

For people loading the tables with the AUDU, you should grant the EXECUTE privilege on the plan of the AUDU utility.

Access to the audit trail table works differently for LOG-ASYNC Selector-only sites that do not have DB2 installed. See*IMS DataPropagator for z/OS Reference* for details about the audit trail in a LOG-ASYNC Selector-only site.

Comparison of audit and trace information

The audit trail table is a valuable centralized repository of historical information about propagation events. You can use the audit trail table to view information about previous propagation-related events. Audit information differs, however, from trace information. Trace information provides more detailed information.

CCU and the audit trail

When writing to the audit trail, the CCU creates at least two records for each phase of processing. Each CCU phase creates a termination record when the phase completes and detail records during execution. Audit trail records are written to SMF for:

- Initialization phase
- IMS read phase for the hashing technique
- DB2 read phase for the hashing technique
- Read phase for the direct technique
- Hash sum compare phase
- Compare phase, when errors are encountered with the hashing technique
- Error location phase
- Final record, written at CCU completion

At completion of its run, the CCU creates a final record for each propagation request. The final record shows inconsistencies between the copies during CCU processing. You can use the information written by the CCU to help resolve inconsistencies between copies of the data.

Monitoring consistency with the CCU

By monitoring propagated data for consistency between IMS databases and DB2 tables you ensure the usefulness of propagated data. Run the CCU periodically to verify data consistency.

If you find data inconsistencies, check the CCU print output or the audit trail for a description of the inconsistencies. The descriptions contain the identifier of the propagation requests that were checked. With these propagation request identifiers, you can query the message table to determine if any warning messages were

written by the MVG when the propagation requests were created. You can then resolve inconsistencies due to propagation failures and erroneous propagation requests.

See Chapter 14, “Verifying Data Consistency (CCU),” on page 203 for a more information on how you can use the CCU.

Monitoring propagation with the message table of the IMS DPROP directory

The message table contains warning messages issued by the MVG during creation of propagation requests. You can use the message table to analyze propagation failures for a specific propagation request for specific data.

If the MVG does not encounter problems when a propagation request is generated, the message table contains no messages. If warning level diagnostic messages are issued, the message table contains one or more rows that give the propagation request identifier, message number and text issued by MVG, and the names of the database, segment type, and DB2 table. Error level messages cause the generation of a propagation request to fail, and no messages are placed in the message table.

You cannot use the message table for LOG-ASYNC Selector-only sites.

Related Reading: Refer to the *IMS DataPropagator for z/OS Reference* for more information on the message table.

Chapter 16. Performance and monitoring

IMS DPROP improved asynchronous propagation performance in the Selector.

- Code path lengths for Selector operations have been reduced, requiring less CPU time
- Automatic log selection has been improved, resulting in reduced time when processing multiple log data sets

Unlike IMS and DB2, IMS DPROP has few components that you can tune. The best performance environment for propagation is one in which IMS and DB2 are performing at optimum levels. You experience less performance impacts during data propagation when databases and table spaces are well organized. For information about tuning IMS and DB2, see the IMS and DB2 libraries, which contain extensive performance and tuning information.

This chapter discusses:

- IMS DPROP performance
- Monitoring propagation

Performance

This section describes some aspects of propagation you should consider to maximize performance in your environment. It is organized by propagation phase, with additional performance considerations given at the end of the section. The sections are:

- Mapping and design phase
- Setup phase
- Propagation phase for LOG-ASYNC and user asynchronous propagation and synchronous propagation performance
- CCU execution

Mapping and design phase

The KEYORDER keyword, used to define propagation requests, can affect performance of the definition process. If you specify ANY, MVG must access the DB2 catalog to determine the proper sequence of the columns of the DB2 primary key. To eliminate this activity, use the proper key sequence (ascending or descending) when you define propagation requests.

Setup phase

You can affect the performance of your system by the way you set up IMS DPROP for IMS-to-DB2 propagation.

The elapsed time required to extract data from an IMS database and load target DB2 tables can be considerable. The elapsed time is directly related to the volume of records being processed. Large buffer pools, cached controllers, and OSAM sequential buffering can reduce elapsed time during the extract. You also have less elapsed time if IMS databases are organized so each database record is placed in as few blocks as possible. Finally, processing the DB2 load in primary key sequence also decreases elapsed time.

You might want to use DataRefresher to *batch* extract requests, allowing multiple segments of the same IMS database be extracted with a single pass through the database.

You also might want to load large DB2 tables using the DB2 LOAD utility. For large tables, using the load utility is generally more efficient than loading the table using SQL insert statements. If you have multiple DB2 tables to load, you might consider loading them in parallel to reduce the elapsed time needed for the extract and load process.

Propagation phase - LOG-ASYNC propagation performance

This section describes the factors that can affect the performance of the Selector, the Receiver, and in some cases, other processes that are active.

Selector

IMS DPROP Version 2 Release 2 improves asynchronous propagation performance in the Selector.

- Code path lengths for Selector operations have been reduced, requiring less CPU time
- Automatic log selection has been improved, resulting in reduced time when processing multiple log data sets

Other factors that affect the performance of the Selector are:

- Selecting multiple groups in parallel

  The major performance factor when running the Selector is the amount of processing involved in reading through the IMS logs to extract the propagation log records relevant to the selected propagation groups. We strongly recommend that all propagation groups that require the same set of IMS logs be processed on the same Selector run so that the Selector reads the set of IMS logs only once.
- Running multiple Selectors in parallel

  Because the SCF is part of the Selector and propagation group recovery mechanism, you can run multiple Selectors in parallel only if each Selector uses its own SCF and ULR data set. The definitions for the propagation groups, including database, segment, field, and subsystem definitions, must be created in the Selector's SCF.
- Using the modified IMS archive job

  SLDS logs contain many record types that are not relevant to data propagation, however, the Selector must process all the record types in order to extract the propagation log records.

  You can reduce the amount of Selector processing required, by modifying the IMS archive job to create, populate, and register CDCDSs. Each time an OLDS is archived into a SLDS, the modified IMS archive job creates a CDCDS and writes all propagation log records in the job stream to it. A separate step in the modified IMS archive job registers the CDCDS to DBRC. See "Modifying the archive JCL to create CDCDSs" on page 227 for details of how to implement the modified IMS archive job.

  During its DBRC phase, the Selector requests details of all SLDSs and CDCDSs that are relevant to the current execution. If a relevant CDCDS is located, it is used instead of the corresponding SLDS. You reduce the Selector processing time but might have increased time to archive an OLDS.

See “Criteria for deciding whether to use SLDSs or CDCDSs” on page 220 for details of the factors you should be aware of when deciding the type of log the Selector is to use.

- Placement of data sets, and access to tape or cartridge units

  The Selector processes log data sets in chronological order. Any logs IMS is writing to simultaneously are opened and read by the Selector in parallel. Each log is dynamically allocated and opened as soon as it is required, and closed and deallocated as soon as all necessary records are read. In environments where log data sets are held on tape or cartridge, you require minimal tape unit access. However, depending on the IMS sources of the groups being selected, the Selector might require a number of tape units.

  For example, if all databases in the selected groups are updated exclusively by one IMS subsystem, then the logs for that subsystem are created sequentially by IMS, and the Selector reads them sequentially and one at time; only one tape unit is required for log reading. However, if the set of databases associated with the selected propagation groups are updated by various IMS subsystems or batch regions, then the associated logs might have been created in parallel, and the Selector reads them in parallel. The number of tape units required for log reading equals the number of logs that IMS writes to at one time within the propagation group selection time span.

  Additionally, you can create PRDS data sets on tape, if the amount of data being propagated is large, or if you need to physically transport data between Selector and Receiver sites. Records are written to the PRDS data sets as they are encountered on the IMS logs. Generally, assuming that the start and stop specifications for the selected propagation groups are similar, PRDS data sets need to be allocated and open during most of the Selector run; therefore a separate tape unit is required for each PRDS during the Selector run.

  We recommended that you use the IMS archive exit process to create CDCDS logs on DASD that you can input directly into the Selector. In single-site environments, or multi-site environments using either DASD sharing or network transportation mechanisms, and where the amount of data to be propagated is not prohibitive, create PRDS data sets on DASD. In multi-site environments where you must physically transport data, create PRDS data sets on DASD, and then copy the data to tape for transportation.

- Frequency of selection

  DB2 application requirements determine how often the propagated data should be applied to the target tables. Options range from propagation where the Selector and Receiver are both run after each IMS archive or batch application termination, to propagation after long-term accumulation of PRDS data sets, with batched Receiver runs.

  Or, as a compromise, you can have the modified IMS archive job submit a low-priority propagation job that runs the Selector and the PRU to the internal reader. Because the job runs after every archive, it only processes one CDCDS or SLDS.

  You can start the job to run the Receiver whenever you want to apply the changes to the DB2 tables. If the time interval between Receiver runs is long, causing a large backlog of propagation data to be applied, you might want to extract the entire IMS source data, and apply the extract to the target DB2 tables.

- Database-, field- and segment-level sensitivity

  You can use sensitivity definitions to reduce the amount of data written to PRDS and passed to the Receiver for application to DB2.

The increase in Selector processing required to test log records for segment sensitivity is insignificant. Therefore, segment-level sensitivity is recommended for all propagating segments, except where that sensitivity is specifically not desired.

Field sensitivity testing, however, can result in performance degradation in the Selector. Generally, if the number of updates made to the non-sensitive part of a segment is large in proportion to the number of updates made to the sensitive part of the segment, then field-level sensitivity is recommended. If the opposite is true, or if a large percentage of the segment is defined as field-sensitive, then segment-level sensitivity might be more suitable.

The Group Unload utility always defines sensitivity at the lowest level possible. If you plan to propagate full segments or databases, change the control statements produced by the Group Unload utility so that you are using segment- or database-level sensitivity instead of field-level sensitivity.

In conjunction with segment or field sensitivity in the SCF, specifying the EXIT=(...,LOG) option in the IMS DBD at the segment level causes logging to be done only for certain segment types, instead of for all segment types within a database. Log record volume is reduced if only a number of segments within a database are being propagated.

- Checkpoints in IMS batch applications

  The Selector must hold all log records for uncommitted UOW in memory until either the UOWs are committed, or the Selector stop time is reached. The number of log records could be large, if the logs are created by batch applications that do not commit on a regular basis.
- Cleanup RECON regularly

  The Selector calls DBRC to retrieve information about all log and CDCDS data sets recorded in RECON. Deleting inactive log and CDCDS data set information from RECON can significantly decrease Selector initialization time.

Criteria for deciding whether to use SLDSs or CDCDSs

The Selector supports the processing of only System Log data sets (SLDSs) and Changed Data Capture data sets (CDCDSs) that are catalogued.

It is strongly recommended that you use the modified archive JCL to create and populate CDCDSs to improve performance. The archive JCL can filter out any records that have been specified for propagation, write them to CDCDSs, and register the CDCDSs to DBRC.

The Selector can then use the CDCDSs, which contain only propagation log record, instead of the corresponding SLDSs. If this extends the time needed to archive the OLDS, switch back to using SLDSs.

To determine whether or not you should use the modified IMS archive job, consider:

- Time constraints on the IMS archive process
- Time constraints on the Selector
- Relative efficiency of the modified IMS archive job in your environment, which is determined by the number of propagation log records written to the log compared to the total number of records written to the log

Keep in mind that:

- You must eventually process the non-propagation type log records; you must decide whether it is more suitable to process at archive time or at Selector run time.

- If you use SLDSs, the Selector run time is increased because it must read all records in the logs to locate all propagation log records.
- If you use CDCDSs, the archive job run time is increased because it must write all propagation log records to the CDCDSs.
- You can limit the number of propagation log records written to the log by specifying the EXIT keyword for only the segments that you source data from, instead of the complete IMS database.

Refer to “Modifying the archive JCL to create CDCDSs” on page 227 for details of how to modify the archive JCL. The following scenarios illustrate the trade-offs between SLDSs and CDCDSs for various ratios of propagation log records and total records written to the logs.

Scenario 1: The number of propagation log records is very small compared to the total number of IMS records written to the log. It is more efficient to use CDCDSs because:

- The Selector run time is greatly reduced, because the Selector needs to read only a small number of propagation log records from the CDCDSs instead of a comparatively large number of log records from the SLDSs.
- The archive job run time is only slightly increased, because the archive job has to write only a few propagation log records to the CDCDSs.

Scenario 2: The number of propagation log records is almost the same as the same as the total number of IMS records written to the log.

It is more efficient to use SLDSs because:

- The Selector run time is only slightly greater for SLDSs than for CDCDSs, because the number of propagation log records in the CDCDSs is almost the same as the total number of log records in the SLDSs.
- The archive job run time is greatly increased, because the archive job has to write a very large number of propagation log records to the CDCDSs.

Scenario 3: The number of propagation log records is moderate, between Scenario 1 and 2.

Weigh the trade-off between the reduced Selector run time required for CDCDSs and the reduced archive job run time required for SLDSs.

When deciding whether to use CDCDSs or SLDSs, consider:

- In an online IMS environment, IMS writes a substantial number of log records that are not required for propagation (for example, records associated with processing message queues, and recovery). This reduces the ratio of propagation log records to total log records.
- Propagation log records are written to only the logs for UPDATE to IMS databases; READ operations do not generate log records. You can limit the amount of UPDATE logging by specifying the most appropriate options on the EXIT keyword for the DBDs and only specifying the keyword for the segments that are required for propagation.

Receiver

View the Receiver as a DB2 application. The performance aspects of the Receiver are the same as any user-written application, which inserts a similar amount of data into the target tables. The same database organization criteria such as the use of table spaces and indices apply.

- Splitting groups over multiple Receivers

  You can split a single propagation group over a number of Receivers, so that each Receiver applies a subset of the propagation requests for a group to DB2. By running multiple Receivers in parallel, you reduce the elapsed time for application of data.
- Using the Receiver Commit Count parameter

  By combining IMS source UOW into a larger DB2 UOW, you can improve the performance of the DB2 apply process. Use the Receiver COMMCNT parameter with a value greater than one. Keep in mind, however, because the DB2 UOW is longer, the performance of other DB2 applications which require access to the same DB2 databases might be affected.
- Using DB2 RIRs to avoid propagation of IMS CASCADE DELETEs

  When you include the CASCADE option on the DBD or SEGM macros, you specify logging of cascading IMS deletes. When a physical parent or ancestor segment is deleted, cascading deletes are logged and the resulting log records are selected, written to PRDS, transported and received as part of Data Propagator operation.

  IMS cascade deletes can more efficiently be propagated by specifying NOCASCADE on the DBD or SEGM macro, if a DB2 CASCADE DELETE referential integrity rule (RIR) is created on the target tables. This reduces the processing done by both the Selector and Receiver, and the amount of log data to be transported.
- Avoid PR generation while propagating

  To reduce contention with IMS DPROP directory tables, it is best not to run the MVG against the DB2 system which has an active Receiver running.
- Avoid using DB2 RUNSTATS utility when the Receiver is running

  The DB2 RUNSTATS utility contends with the Receiver. The Receiver waits until RUNSTATS completes before accessing tables.

General suggestions

- Run Receivers regularly, perhaps every PRDS receipt

  The Receiver run time is affected by the size of the PRDS file being processed. To minimize Receiver run time and DB2 table processing, keep the size of the PRDS files small.
- Minimize IMS DPROP directory and target table contention

  The MVG and Receivers contend with each other for access to IMS DPROP directory tables. Avoid accessing target tables when the Receiver is running.
- Tracing

  At some trace level settings, the IMS DPROP components might produce a large amount of trace data. The data can be useful for problem determination, but in a production environment, we recommend that you specify minimum tracing (trace level 1).

  Trace level 2 is useful for tracing the data being propagated.

Propagation Phase: User asynchronous propagation performance

In user asynchronous propagation, if you request that the IMS Asynchronous Data Capture exit write changed segments to the log, then your updating applications are impacted by the additional log records. Writing additional log records is usually minor unless your application's performance is constrained by the amount of IMS logging. Examine the various alternatives for externalizing records to determine the best method for your installation. Alternatives might include externalizing:

- The IMS log (OLDS)
- The IMS message queue
- A full function IMS database
- Sequential dependents of a DEDB
- An MVS, or flat, file

SQL call processing affects the receiving program that calls RUP. If you assign control to your own Asynchronous Data Capture exit, your updating applications are impacted by the processing performed in your exit routine. The impact depends as several variables including how many times you update propagated segments and invoke your exit routine.

CCU processing

The sequential processing characteristics of your IMS databases and DB2 tables are the most important factor in CCU performance when you use the hashing technique (without concurrent updates) or the direct technique. To improve sequential processing, you can use large database buffer pools and cached controllers. Databases and table spaces can also be reorganized to improve performance. For IMS databases, OSAM sequential buffering can also improve performance.

If you are using the hashing technique, you can improve performance by splitting the CCU run into several different job steps. A full CCU run using the hashing technique consists of the following phases:

- Initialization
- IMS database read
- DB2 database read
- Hash sum compare
- Compare and error location

You can create job streams for each of the first three phases, but the hash sum compare phase and the compare and error location phase should be run together in a single job stream. You can omit running these two phases if you specified KEYONLY or HASHONLY in the initialization phase.

To minimize the number of work records written by the CCU and improve performance, you can specify HASHONLY.

If you specify KEYONLY in the CHECK statement of the DB2 read phase, the CCU reads the DB2 indexes instead of the table spaces, reducing time. You are only verifying the existence of IMS segments and DB2 rows, and not their content, when you specify KEYONLY. However, if you are submitting the CCU using an HD unload file as a replacement for the regular IMS database, HASHONLY and KEYONLY keywords probably do not save elapsed time in the IMS database read phase of the hashing technique because the CCU needs to sequentially access the HD unload file. Sequential access is probably the most time consuming phase of the CCU

If the CCU encounters consistency errors or if concurrent updating is allowed, a sort is required. The amount of time required for the sort depends on the number of records to be sorted, which in turn depends on the database and table size.

Monitoring propagation

This section describes some of the IMS and DB2 monitoring tools you can use to tune your system and maximize performance.

Although you cannot monitor propagation performance from within IMS DPROP, you can use tools such as the IMS Monitor to determine the amount of time required to service IMS and SQL calls. The IMS Monitor describes in detail where resources are used in IMS calls. To determine where resources are used in SQL calls, use a DB2 monitoring tool such as the DB2 Performance Monitor.

Related Reading: For more information on the IMS Monitor, refer to *IMS/ESA Administration Guide: System* and *IMS/ESA Administration Guide: Database Manager*. Information on how to run the DB2 Performance Monitor is in *DB2PM Report Reference*.

IMSPARS and IMSASAP II are other IMS monitoring tools you might find useful.

Related Reading: Refer to *IMSPARS Program Description and Operation Manual* and *IMSASAP II Program Description and Operation Manual* for more information about these products.

For synchronous propagation, the DB2 EXPLAIN utility can give you information about the access paths selected by propagating SQL calls.

Related Reading: For information on DB2 EXPLAIN, see *DB2 Administration Guide* and *DB2 Utility Guide and Reference*.

The Database Tools product is useful for tuning IMS databases that are involved in data propagation. While no special guidelines can be given for IMS databases involved in data propagation, a well-tuned and efficient IMS database can be propagated with less performance impact than one that is poorly tuned.

Part 5. Appendixes

Appendix A. JCL information

This contains detailed information about JCL changes that you can or must make in the IMS DPROP environment.

- Modifying the archive JCL to create CDCDSs
- JCL changes for synchronous propagation
- JCL changes for DB2

Modifying the archive JCL to create CDCDSs

You can improve Selector performance by modifying the archive JCL to create Changed Data Capture data sets (CDCDSs) and write to the data sets to only the log records that are required for propagation. When you run the Selector, the Selector can use the CDCDSs instead of the corresponding SLDSs.

The sample JCL member, EKYUARCS, shown in Figure 50 on page 228 shows the skeleton archive JCL, with:

- The //CDCDS DD statement, specifying the CDCDS parameters
- The //SYSIN DD statement, specifying the IMS record types that are to be written to the CDCDS
- The job step to register the CDCDS to the RECON data sets

Before submitting the JCL, you must replace 1111111 with the high level qualifier of the IMS DPROP RESLIB.

You must specify the RECON data sets in both steps shown in Figure 50 on page 228, if they are not being dynamically allocated. You must use the IMS DPROP program EKYB800X to register the CDCDSs in the RECONs with IMS DPROP-specific parameters. The Selector uses these parameters during CDCDS log location.

```
//*********************************************************************
//**   EKYUARCS - IMS ARCHIVE JCL TO CREATE CDCDS                    */
//*********************************************************************
//**                                                                 */
//**        LICENSED MATERIALS - PROPERTY OF IBM                     */
//**                                                                 */
//**        5655-E52 (C) COPYRIGHT IBM CORP. 1994.                   */
//**                                                                 */
//**        SEE COPYRIGHT INSTRUCTIONS                               */
//**                                                                 */
//*********************************************************************
//**          1111111    -DPROP RESLIB HIGH LEVEL QUALIFIER          *
//*********************************************************************
//ARCHIVE  EXEC PGM=DFSUARC0,PARM='%SSID,DBRC=YES'
//SYSPRINT DD SYSOUT=*
%SELECT OLDS(%SSID,(%DDNAMES))
//%OLDSDDN DD DSN=%OLDSDSN,DISP=SHR
%ENDSEL
//DFSSLOGP  DD DSN=IMSESA.SLDSP.%SSID.D%ARDATE.T%ARTIME.V%ARVERS,
//             DISP=(NEW,CATLG),SPACE=(CYL,(15,2),RLSE)
//CDCDS     DD DSN=IMSESA.CDCDS.%SSID.D%ARDATE.T%ARTIME.V%ARVERS,
//             DISP=(NEW,CATLG),SPACE=(CYL,(15,2),RLSE)
//SYSIN     DD *
COPY DDNOUT1 (CDCDS) -
 RECORD (O(5) T(X) V(9904) L(2) C(E)) -
 RECORD (O(5) T(X) V(9928) L(2) C(E)) -
 RECORD (O(5) T(X) V(9930) L(2) C(E)) -
 RECORD (O(5) T(X) V(9934) L(2) C(E)) -
 RECORD (O(5) T(X) V(37) L(1) C(M)) -
 RECORD (O(7) T(X) V(40) C(ETY)) -
 RECORD (O(5) T(X) V(5937) L(2) C(M)) -
 RECORD (O(37) T(X) V(20) C(ETY)) -
 RECORD (O(5) T(X) V(5938) L(2) C(M)) -
 RECORD (O(37) T(X) V(20) C(ETY)) -
 RECORD (O(5) T(X) V(38) L(1) C(M)) -
 RECORD (O(8) T(X) V(40) C(ETY)) -
 RECORD (O(5) T(X) V(41) L(1) C(M)) -
 RECORD (O(26) T(X) V(01) C(ETY))
  /*
//*
//*********************************************************************
//* THIS STEP REGISTERS THE CDCDS IN THE RECON DATASETS
//*********************************************************************
//NOTIFY   EXEC PGM=EKYB800X,PARM='%OLDOTIM',COND=(4,LT)
//EKYRESLB  DD DISP=SHR,
//         DSN=1111111.EKYRESLB
//SYSPRINT  DD   DSN=&&SYSPRINT,UNIT=SYSDA,
//          SPACE=(TRK,2)
//SYSIN     DD  DSN=&&TEMP,UNIT=SYSDA,SPACE=(TRK,2),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
//SLDS      DD  DSN=*.ARCHIVE.DFSSLOGP,DISP=SHR
//CDCDS     DD  DSN=*.ARCHIVE.CDCDS,DISP=SHR
//SELPRINT  DD  SYSOUT=*
```

Figure 50. EKYUARCS

Appendix B. LOG-ASYNC Propagation data sets and storage requirements

This chapter contains information on the following LOG-ASYNC propagation data sets:

DFSLOGP	IMS System Log
CDCDS	Change Data Capture
EKYPRDS	Propagation Request
EKYSCF	Selector Control File
EKYULR	Uncommitted Log Record
EKYPREG/EKYREGIN	PRDS Register Input
EKYGRPD	PR Group Definition

DFSSLOGP

The IMS System Log data set (SLDS) is produced by the IMS Archive utility and contains all the IMS log records produced by the online system. You can use SLDS as input to any of the IMS or IMS DPROP programs that require IMS log records. IMS DPROP also uses SLDSs in DBRC.

The Selector uses the DBRC SLDS entries to determine what input files are required. Even if the SLDSs do not contain any propagation log records, their DBRC entries are used to determine the input needed because propagation is controlled by start and stop timestamps. If the last propagation record for a particular database was created 4 weeks ago, all the SLDS entries since that time must be examined to determine their relevance. Only by considering these entries can the Selector determine which SLDSs it must allocate.

When you no longer need to use the SLDS for recovery you can delete SLDSs and their DBRC entries either individually or as a group using the DELETE.LOG INACTIVE function of DBRC. Do not delete the SLDSs until you sure they are not needed. Usually you keep SLDSs until you delete the Image Copy data set to which they apply.

CDCDS

This is an optional data set that is created using the IMS Archive utility's COPY function and contains the IMS log records that are required for propagation. The log records are identified using a set of SELECT statements (see Figure 50 on page 228). Use this data set as input to the Selector in place of the associated System Log data set (SLDS) to reduce the time taken by the Selector to read the required log records.

To produce the CDCDS data set, modify the IMS Archive JCL skeleton:

1. Add a DD statement with the following parameters:

   DDNAME For example, CDCDS

   DSN= Name the data set. In this book, CDCDS is used.

   SPACE= See the following discussion on volume and allocation.

DISP=(NEW,CATLG)
See the following discussion on catalogs and registration.

UNIT= If required

2. Add to the SYSIN input stream:
   - COPY DDNOUT1 (*ddname*)

     Where *ddname* is the DD name used in step 1 on page 229
   - The SELECT control statements supplied in member EKYUARCS of the sample JCL library
3. Add a step to run the CDCDS Registration utility.

The CDCDS Registration utility creates entries in DBRC (recons) that are used by the Selector to determine which input files to use. When a CDCDS is available, it is dynamically allocated in place of the corresponding SLDS. If the CDCDS is not registered or is not available when the Selector runs, the SLDS is used instead. We recommend that you check the Selector job output to ensure that the CDCDS is being used correctly with the SLDS.

Note: CDCDSs must be cataloged, so that the Selector can dynamically allocate them. If any CDCDS is not cataloged, the Selector is unable to allocate it, because the volume used for registration is not the volume on which the data set resides.

In addition to registering the CDCDSs to DBRC, you must also remove entries that are no longer required. You can use IMS DPROP's CDCDS Deletion utility. The utility automatically deletes any CDCDS entries in DBRC that are older than a specified time. The time is calculated by subtracting the number of days specified in the PARM string from the current timestamp.

When specifying a delete time, consider the recovery situations that might be required if propagation fails at the Receiver site or at a later time. Specify a value that allows for reruns of the propagation. Or, provide additional backups of the relevant files so that you can retrieve the necessary data.

The CDCDS Deletion utility deletes only DBRC entries, not the data sets. Delete data sets manually or by setting a suitable limit value in the GDG index. If you use the GDG index make sure that the value selected does not cause the utility to delete data sets still required for propagation or maintain data sets of DBRC entries that have been deleted.

Although this data set is optional, using it can significantly reduce the number of log records that the Selector has to read and can, therefore, reduce run time.

The data set attributes are:

DSORG
Sequential.

RECFM
VB.

LRECL
As for SLDS.

BLKSIZE
As for SLDS.

SIZE This will depend on the amount of propagation and the frequency of change. As a guide, use half the size of the SLDS and include the RLSE parameter to avoid over-allocation.

EKYPRDS

The Selector creates the PRDS. The PRDS contains records that are forwarded to the Receiver for processing. One PRDS is created for each of the propagation groups defined in the SCF. It is recommended that you define PRDSs using GDGs so that each PRDS is written to a new data set and each propagation group uses a separate GDG index. If you use GDGs, you need a procedure for deleting old PRDSs. You can automate the procedure using the LEVEL function of the GDG, but you must ensure that the value used address all possible recovery situations. Or you can delete PRDSs manually. In either case, you should also remove the corresponding entry in the PRDS Register file so that the register and existing data sets remain consistent.

You might also use the 'SELECT=ALL' function in the Selector (to select every register group) and include in the JCL a PRDS DD statement for every registered group. Without the statement, propagation abends when the Selector attempts to open the DD name of each group registered in the SCF. If you delete a group from the SCF, then you should remove the corresponding DD statement from the JCL to prevent the PRDS being catalogued. Otherwise, a null file is catalogued, which can cause problems. However, registered groups do not incur a problem for PRDSs because these PRDSs are always opened and closed by the Selector even if there are no propagation log records for that group.

Do not use tapes for PRDSs because each PRDS must be opened as part of PRDS registration and tapes increase time taken. Also, the PRU runs as a terminal monitor program (TMP) and you would need TSO MOUNT authorization in order to access the PRDS on tape.

The data set attributes are:

DSORG
Sequential

RECFM
VB

LRECL
32756

BLKSIZE
32760

SIZE Depends on the size of each database segment and the number of changes that occur in each segments. Other factors to consider are the commit frequency and amount of compression that can be achieved.

You can use the IMS Log Select/Print utility to determine the number of propagation log records present in a SLDS/CDCDS. Use the number to determine the overall size of the PRDS. You would still need to determine how to appropriate PRDSs across the various groups. The number of propagation log records might vary within a given time period (for example, you might have more propagation log records at month end, year end, and so on).

Considering these factors, we recommended that you allocate each PRDS with a primary space of at least 50% of the input CDCDSs or 20% of the input SLDSs and a secondary space of 10% of the primary space. Each allocation should also include the RLSE parameter so that unused space is given back to the system.

EKYSCF

The EKYSCF file is created and initialized when you generate IMS DPROP during installation. The file is used by the Selector to store all the information required to control propagation. Information includes details of the data to be propagated (DBs, segments, fields), any propagation groups with their start and stop times and the current status of each group, for example, ACTIVE and SUSPENDED.

Because EKYSCF is a VSAM data set, you must perform regular maintenance on the file to preserve the percentage of free space and minimize the likelihood of CI and CA splits. You can use REPRO to unload and reload the file, reestablish the free space specification and regulate the number of used extents.

The data set attributes are:

DSORG
VSAM

RECFM
KSDS

LRECL
142 (max)

BLKSIZE
18,432 (3390 Model 2)

SIZE Depends on the amount of propagation and the number of groups and propagation requests defined. Use the value supplied in the sample code and monitor the data set at regular intervals to determine frequency of use.

EKYULR

The EKYULR file is created when you generateIMS DPROP during installation. The Selector uses EKYULR to store IMS propagation log records that do not have an associated commit or abort log record and, therefore, cannot be passed to the Receiver or discarded. The records are stored in the ULR until the associated commit or abort record is processed. Then the records are either discarded or transferred to the relevant PRDS for forwarding to the Receiver in the normal way.

You might encounter situations where the commit record for an associated set of changes is not produced. For example, an IMS system has to be *cold* started after an outage because of I/O errors on the active OLDS. In this case, the log records remains on the ULR until you delete it. You can perform a GROUP delete for the appropriate group and then reinstating it. Or you can initialize the ULR when it only contains those records, for example, after all propagation log records have been processed following a normal IMS closedown and when all groups have a STOPTIME= INTERIM specified.

The data set attributes are:

DSORG
VSAM

RECFM
KSDS

LRECL
Average = 1024, Maximum = 31768

BLKSIZE
18,432 (3390 Model 2)

SIZE Depends on the amount of propagation and the frequency of uncommitted log records. Use the value supplied in the sample code and monitor uncommitted log records at regular intervals to determine frequency of use.

EKYPRREG/EKYREGIN

The EKYPRREG/EKYREGIN file is created by the Selector and contains control statements used by the PRDS Registration utility (PRU). The control statements contain a registration command for each PRDS and facilitate PRDS registration. You can use the control statements as input to the PRU at the Receiver site to maintain the PRDS Register.

Create the EKYREGIN data set once and then include it in the Selector JCL with a DISP=OLD. Each time the Selector runs, a new set of PRDS registration statements are created. This may be unacceptable if the file has not been successfully processed by the time of the next Selector run. If you use DISP=MOD, have a procedure to remove unwanted entries so that you control the size of the data set and eliminate the need for the PRU to process all entries and then select only entries that are relevant. We recommend that you periodically delete and reallocate the data set ensuring that registration statements still needed are either reinstated or performed manually.

The data set attributes are:

DSORG
PS

RECFM
FB

LRECL
80

BLKSIZE
Any multiple of the record size as required.

SIZE Depends on the number of PRDSs produced and the frequency with which the cleanup is run. As a guideline, use the supplied values and monitor the data set at regular intervals to determine the actual size needed.

EKYGRPD

The EKYGRPD file is created by the Group Unload utility and contains information about propagation requests and groups that are selectively extracted from the IMS DPROP directory tables at the Receiver site. Define the information in the SCF using the SCF Compare and Apply utility so that the two environments are consistent.

The data set attributes are:

DSORG
PS

RECFM
FB

LRECL
80

BLKSIZE
Any multiple of the record size as required.

SIZE Depends on the number of groups and propagation requests defined. Use the value supplied in the sample and include the RLSE parameter to avoid over allocation.

Appendix C. Converting PRTYPE=F into PRTYPE=E propagation requests

In IMS DPROP Version 2, the rules and requirements for PRTYPE=E are very similar to the R1 rules for PRTYPE=Fs. (See “Propagation requests and selecting PRTYPEs” on page 11.) PRTYPE=E rules are somewhat more restrictive then the R1 rules for PRTYPE=F so that PRTYPE=E can support DB2-to-IMS propagation.

You can convert a R1 PRTYPE=F into a R2 PRTYPE=E. Conversion is done by changing the PRTYPE parameter and by recreating the propagation request. You might want to convert PRTYPEs if you plan to eventually implement DB2-to-IMS propagation. When converting, note that R2 rules for PRTYPE=E are more restrictive than R1 rules for PRTYPE=F.

The rules for converting a R1 PRTYPE=F into a R2 PRTYPE=E are:

- With a PRTYPE=E, you must not propagate an IMS field to more than one column of the same table if the field is:
  - The IMS key field
  - Part of the IMS key field
  - Mapped to the DB2 primary key

  IMS DPROP does not impose the same restriction on other IMS fields. However, with DB2-to-IMS synchronous propagation, we do not recommend that you propagate the same field to more than one column. Doing so can result in inconsistent data.
- IMS DPROP limits the number of PRTYPE=Es that can propagate a particular segment type.

  With a few exceptions, you cannot create multiple PRTYPE=Es propagating the same segment to and from multiple tables. The exceptions are:
  - IMS segments containing internal segments that are propagated with mapping case 3 propagation requests. Each internal segment can be propagated at most by one PRTYPE=E. The segment containing the internal segment can be propagated by another PRTYPE=E.
  - IMS segments propagated by multiple PRTYPE=E, if all these propagation requests specify a WHERE clause. One segment occurrence should not satisfy the WHERE clause of more then one of these propagation requests. All of the propagation requests must belong to the same mapping case and have the same mapping direction.

  IMS DPROP allows you to propagate the same segment both with PRTYPE=E and with one or multiple PRTYPE=Ls.
- For PRTYPE=E, extension segments of a mapping case 2 propagation request:
  - Must not have an IMS key field
  - Must not have dependent segments propagated by PRTYPE=Es
- For IMS unidirectional logical relationships, the IMS delete rule for the logical parent segment must be PHYSICAL. PRTYPE=F allows both a PHYSICAL and LOGICAL delete rule.
- For paired logical child segment types propagated by PRTYPE=Es, IMS DPROP rejects propagation request definitions if both paired segments are propagated by propagation requests. If the pairing is VIRTUAL, then the physical child should be propagated. If the pairing is PHYSICAL, then the child with propagated physical dependent segments should be propagated.
- If implementing a DB2 RIR matching an IMS logical parent/child relationship, observe the following rule:

Match an IMS PHYSICAL delete rule of the logical parent with a DB2 delete rule of ON DELETE RESTRICT PRTYPE=F allows both ON DELETE RESTRICT and ON DELETE CASCADE.

- If performing DB2-to-IMS or two-way synchronous propagation, the physical parent and ancestors of a propagated segment must also be propagated in the same direction.

If you convert a PRTYPE=F into a PRTYPE=E, then you should extend the logic of your Segment and Field exit routines to support DB2-to-IMS mapping, even if you perform only IMS-to-DB2 propagation. Extending the exit routine logic enables CCU to call your exit routines to perform DB2-to-IMS mapping.

Appendix D. Example of LOG-ASYNC propagation

This appendix describes some of the commands and control statements that you might use in a simple LOG-ASYNC propagation scenario. The sequence of tasks is the same as in Chapter 5, “Setting up your systems for LOG-ASYNC propagation,” on page 89, except in this scenario you modify the IMS Archive JCL early in the process. Assume that:

- You want to propagate changes from the following IMS segments to the indicated DB2 tables:

  From To

  SEG01
  TAB01

  SEG02
  TAB02

- SEG01and SEG02 are part of IMS database DBD1 in subsystem SSID01
- TAB01 and TAB02 are in DB2 subsystem DSNA.
- Propagation request, PR01 is used to map SEG01 to TAB01
- Propagation request, PR02 is used to map SEG02 to TAB02
- Both PR01 and PR02 are processed by Receiver REC001 and are grouped together into propagation group GROUP01 for LOG-ASYNC propagation

The tasks you must perform to implement asynchronous propagation are:

1. Set up IMS to facilitate LOG-ASYNC propagation, as follows:
   - Register each database that is to source propagation data with DBRC.
   - Specify, in the DBD of each database, that updates are to be logged and regenerate the DBD and, for online, the ACB.
   - Optional: Modify the IMS Archive JCL so that records used during propagation are written to separate data sets (CDCDSs).
2. Create the mappings between IMS and DB2, using either:
   - DataRefresher, which creates an extract request that maps from IMS to DB2, and calls IMS DPROP (MCE and MVG) to generate a propagation request from the extract request.
   - MVG input tables, which puts mapping definitions into these tables using either IMS DPROP ISPF panels or user-written SQL. From these input tables, the IMS DPROP MVGU and MVG generate the propagation requests.
3. Define propagation groups as collections of one or more related propagation requests, by:
   - Creating various Receivers to process the propagation groups.

     A Receiver can only process one propagation group, but a propagation group can be processed by many Receivers, enabling the propagation of different propagation requests in parallel.

Control statement for the SCU CREATEREC RECNAME=REC001,GROUP=GROUP01;

   - Assigning propagation requests to the various Receivers. Each propagation request can be assigned to only one Receiver.

control statements for the SCU ASSIGNPR PR=(PR01),RECNAME=REC001; ASSIGNPR PR=(PR02),RECNAME=REC001;

For each propagation group, establish and store in a VSAM file known as the Selector control file (SCF), the IMS source (subsystem/database/segment/field) of all propagation requests within the propagation group.

a. Run the Group Unload utility

control statement input in EKYRIDS SELECT GROUP=GROUP01;

---

records generated in EKYGRPD GROUP GROUP=GROUP01 ; DATABASE GROUP=GROUP01, DBD=DBD1 ; SEGMENT GROUP=GROUP01, DBD=DBD1, SEG=SEG01 ; FIELD GROUP=GROUP01, ... SEGMENT GROUP=GROUP01, DBD=DBD1, SEG=SEG02 ; FIELD GROUP=GROUP01, ...

b. Run the SCF Compare utility

records input in EKYGRPD (altered to remove field sensitivity) GROUP GROUP=GROUP01 ; DATABASE GROUP=GROUP01, DBD=DBD1 ; SEGMENT GROUP=GROUP01, DBD=DBD1, SEG=SEG01 ; SEGMENT GROUP=GROUP01, DBD=DBD1, SEG=SEG02 ;

---

records generated in EKYSIDS ADDGROUP GROUP=GROUP01 ; ADDDBASE GROUP=GROUP01, DBD=DBD1 ; ADDSEGM GROUP=GROUP01, DBD=DBD1, SEG=SEG01 ; ADDSEGM GROUP=GROUP01, DBD=DBD1, SEG=SEG02 ;

c. Run the SCF Apply utility

records input in EKYSIDS (which can be altered) ADDGROUP GROUP=GROUP01 ; ADDDBASE GROUP=GROUP01, DBD=DBD1 ; ADDSEGM GROUP=GROUP01, DBD=DBD1, SEG=SEG01 ; ADDSEGM GROUP=GROUP01, DBD=DBD1, SEG=SEG02 ;

---

updated SCF records 0200 DBD1 0300 GROUP01 0302 GROUP01 DBD1 0303 GROUP01 DBD1 SEG01 0303 GROUP01 DBD1 SEG02

4. Establish a base point from which to start propagation, as follows:
   a. Quiesce the IMS databases from which propagation is required.

control statement for the SCU READON DBD=DBD1;

b. For each database, store in the SCF the time at which the database is quiesced (as the start times for propagation groups).

control statement for the SCU CREATETSM QUIESCE,DBD=DBD1,ID=T100294A ;

SCF records updated 0202 DBD1 timestamp T100294A

---

c. Either now or before you run the Selector, assign the times at which the databases were quiesced as propagation group/database start times.

control statements for the SCU ASSIGNTSM GROUP=GROUP01,DBD=DBD1,TIME=TSM, ID=T100294A;

SCF records updated 0302 GROUP01 DBD1 timestamp T100294A newstart_on

d. Optional: For each propagation group, store in the SCF the time at which to stop propagation from all databases that are sources for that propagation group.
e. Perform a full extract of data from the IMS databases and load the data into the DB2 target tables. We also recommended that you establish a

recovery point at this time, by taking image copies of the IMS and DB2 databases, as well as a backup of RECONS, SCF and the ULR data sets.

f. Restart the IMS databases.

Control statement for the SCU READOFF DBD=DBD1;

5. Update the SCF to contain a list of all sub-system identifiers (SSIDs) from which propagation can occur.

   Run the SCF Apply utility

control statement input in EKYSIDS ADDSSID SSID=SSID01 ;

updated SCF records 0101 SSID01

6. Run the Selector, which:
   a. Uses the information stored in the SCF to select records for committed units of work (UOW) from the IMS logs for one or more propagation groups in a specific time range (starting from the database quiesce time when the full extract was done)
   b. For each propagation group, writes the selected records to a sequential file known as a propagation request data set (PRDS)
   c. Writes uncommitted log records to the ULR data set.

Selector control statement input in EKYSIDS SELECT GROUP=GROUP01;

dd statement in Selector JCL for PRDS for GROUP01 (first step) //GROUP01 DD DSN=prds_gdg_name(+1),DISP=(NEW,PASS), // SPACE=(32760,(size),RLSE)

dd statement in Selector JCL for PRDS for GROUP01 (second step) //GROUP01 DD DSN=prds_gdg_name(+1),DISP=(OLD,CATLG)

statement generated in EKYPRREG REGISTER DSN=fully qualified PRDS name

updated SCF records 0302 GROUP01 DBD1 timestamp T100294A newstart_off 0305 GROUP01 timestamp tsm_id selector_complete sequence_number

7. Optional: Unload the IMS databases into an HD unload file for consistency checking later. See Step 11 on page 240.
8. If the IMS and DB2 systems are on different MVS images, without shared DASD, transport the PRDSs to the DB2 MVS image.

Assume name of PRDS received at target site is HLQ.PRDS.GROUP01.DAY1

9. Use the PRDS Registration utility (PRU) to register the PRDSs in the PRDS register table.

Control statement input on EKYREGIN REGISTER DSN=HLQ.PRDS.GROUP01.DAY1; or REGISTER LEVEL=HLQ.PRDS.GROUP01.*;

10. Run the Receiver, which:
    a. Locates the next PRDS to process for a propagation group
    b. Reads the log records in the PRDSs
    c. Uses the RUP to apply the updates to the DB2 target tables, based on the propagation requests assigned to it as defined in Step 2 on page 237

Receiver control statements input on EKYRIDS RECEIVER NAME=REC01,DB2SSID=DSNA, . . . ; PRDS GROUP=GROUP01, . . . ;

11. Optional: Use the CCU to check the consistency of the IMS and DB2 copies of the data, using the HD unload file created in Step 7 on page 239

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 iF 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
J46/G4
555 Bailey Avenue

San Jose, CA 95141-1003
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:

fl (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. fl Copyright IBM Corp. _enter the year or years_. 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 chapter or section or by the following marking:

Notice
This chapter documents general-use programming interface and associated guidance information.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A complete and current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml.

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

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 datasets. 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 documentation referenced in this book

The following titles are referred to in this documentation. If not referred to, they might be helpful in understanding administration tasks:

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

SC26-3269 *DB2 for OS/390 Data Sharing: Planning and Administration*, Version 4

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 5

SC26-3062 *IMS/ESA Application Programming: DL/I Calls*, Version 4

SC26-8015 *IMS/ESA Application Programming: Database Manager*, Version 5

SC26-3065 *IMS/ESA Administration Guide: Database Manager*, Version 4

SC26-8012 *IMS/ESA Administration Guide: Database Manager*, Version 5

SC26-3064 *IMS/ESA Customization Guide*, Version 4

SC26-8020 *IMS/ESA Customization Guide*, Version 5

SC26-3076 *IMS/ESA System Definition and Tailoring*, Version 4

SC26-8024 *IMS/ESA Installation Volume 2: System Definition and Tailoring*, Version 5

SC26-4329 *IMS/ESA Utilities Reference: System*, Version 4

SC26-8035 *IMS/ESA Utilities Reference: System*, Version 5

SC26-3072 *IMS/ESA Operations Guide*, Version 4

SC26-8029 *IMS/ESA Operations Guide*, Version 5

GC26-8031 *IMS Release Planning Guide*, Version 5

SC26-3075 *IMS/ESA Administration Guide: System*, Version 4

SC26-8013 *IMS/ESA Administration Guide: System*, Version 5

SC26-4627 *IMS/ESA Utilities Reference: Database*, Version 4

SC26-8034 *IMS/ESA Utilities Reference: Database Manager*, Version 5

GN28-1257 *OS/390 MVS Application Development Guide*

GC28-1473 *OS/390 MVS JCL User\x27–s Guide*

GN28-1427 *OS/390 MVS Initialization and Tuning*

GC28-1449 *OS/390 MVS Setting up a Sysplex*

GC28-1208 *OS/390 Parallel Sysplex Overview: An Introduction to Data Sharing and Parallelism*

GC26-3398 *An Introduction to DataPropagator Relational*

SC26-3399 *IBM DB2 Universal Database Replication Guide and Reference*

Accessibility titles cited in this book

The following documentation refers to accessibility support:

Accessibility titles

- *z/OS ISPF User's Guide, Volume 1*, SC34-4822
- *z/OS TSO/E Primer*, SA22-7787
- *z/OS TSO/E User's Guide*, SA22-7794

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

X

File Number: 5655-E52

Printed in USA

SC27-1216-02