IBM IMS DataPropagator for z/OS

Administrators Guide for MQSeries Asynchronous Propagation

Version 3 Release 1

SC27-1217-02

IBM IMS DataPropagator for z/OS

Administrators Guide for MQSeries Asynchronous Propagation

Version 3 Release 1

SC27-1217-02

Note!
Before using this information and the product it supports, be sure to read the general information under “Notices” on page 207.

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. The technical changes for this edition are indicated by a vertical bar to the left of a change.

Contents

Figures

About this information

This information describes how to administer IMS™ DataPropagator for z/OS® in MQSeries asynchronous mode.

Administration of IMS DataPropagator includes the design, implementation, and control of data propagation, as well as operation in the data propagation environment. This information describes tasks that an IMS DataPropagator administrator performs.

The information covers MQSeries-based asynchronous propagation (MQ-ASYNC). MQ-ASYNC mode propagates changed data from IMS databases to duplicate copies of these IMS databases or to DB2 tables in both near real-time and in point-in-time.

The information is divided into four parts:

- "Part 1, "Overview of administrative tasks," on page 1," presents key concepts and offers general information about MQ-ASYNC. It also includes a summary list of administrator tasks. Part 1 consists of chapters 1-3.
- "Part 2, "Mapping and design," on page 27," 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 MQ-ASYNC environment. Part 2 consists of chapters 4-6.
- "Part 3, "Setting Up for Data Propagation," on page 99," 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 7-11.
- "Part 4, "Propagating with MQ-ASYNC," on page 157," 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 12-18.

What You Should Know

This book assumes that you understand what data propagation is and the business reasons for propagating data. Information on these topics is in *An Introduction*, GC27-1211. A more technical conceptual look at data propagation is in *IMS DataPropagator for z/OS: Concepts*, SC27-1544.

This book also assumes that you understand IMS, DB2, and DataRefresher concepts and functions.

Key administrative tasks for design, setup and implementation of data propagation are listed in Chapter 2, "Administrative tasks for MQ-ASYNC," on page 23. The order in which tasks are presented is the recommended order but not required.

Terms used in this book

The following terms are synonymous in this book:

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

  In all cases, these terms refer to:

  - Any database you can propagate, except for DEDBs, that is set to READONLY 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

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

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

Using LookAt to look up message explanations

LookAt is an online facility that lets you 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/OS.e 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

These topics provide an overview of the task you perform managing propagation.

Chapter 1. Overview of MQ-ASYNC propagation

This chapter provides an overview of MQ-ASYNC propagation. It explains how propagation works from IMS sources to IMS and DB2 targets using the MQ-ASYNC components of IMS DPROP.

IMS-to-DB2 propagation

Figure 1 shows how MQ-ASYNC propagation works when IMS and DB2 are on different z/OS images. MQ-ASYNC propagation can coexist with LOG-ASYNC propagation.

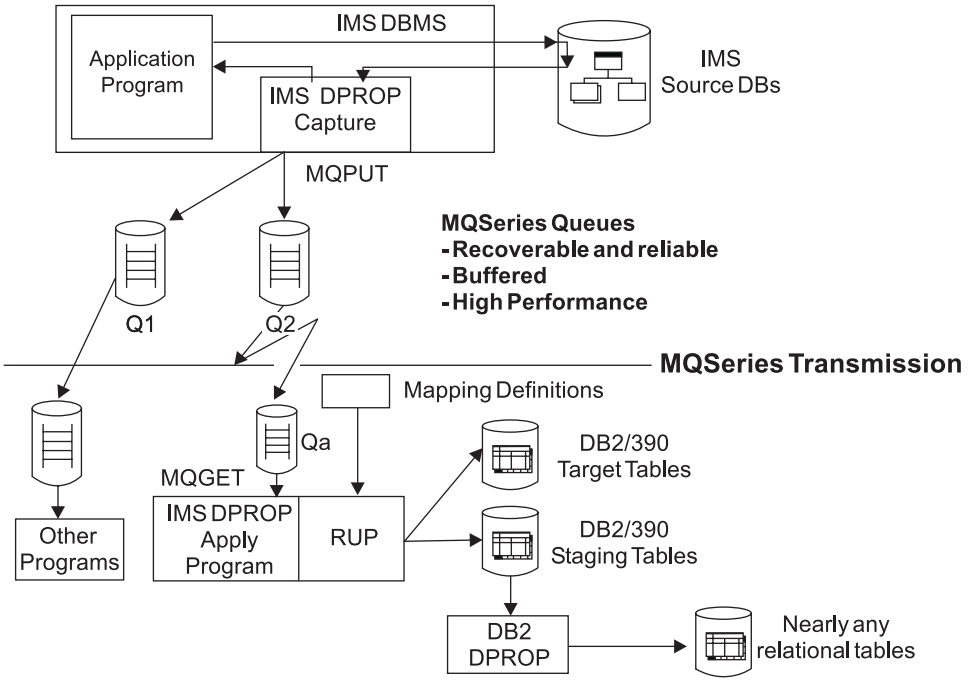


Figure 1. Propagation Phase: MQ-ASYNC Asynchronous IMS-to-DB2

The following list describes Figure 1:

1. When an IMS application changes an IMS segment type to be propagated, DPROP captures the update using a Capture exit routine. This “capture” happens during the unit-of-work (UOW) of the IMS application. Control is passed to the Capture exit routine after the update to the segment is made.
2. The Capture exit routine builds an MQSeries message containing the information about the changed segment and places the message on an MQSeries message queue (labeled Q2 in Figure 1). The Capture exit routine can place the message on multiple queues, as shown in Figure 1.
3. If all the processing performs with no problems (update to the segment, building of the message and placing of the message on the MQSeries queue), the update to the segment is committed. The IMS application’s UOW is now complete.
4. After the segment change is committed, MQSeries transmits the message to another MQSeries queue (labeled Qa in Figure 1) on the system where DB2 is running. This second system is called the target system.

The changed IMS data flows from the Capture component to the Apply program occurrences as part of one or more *propagation data streams* (PRSTREAMs). All changed data contained in one PRSTREAM is transmitted by means of the same MQSeries Queues and is applied in First-In First-Out (FIFO) sequence by the same occurrence of the Apply program.

5. The IMS DPROP Apply program retrieves the message from queue Qa and passes it to the DPROP Relational Update Program (RUP).
6. The RUP queries the mapping definitions that describe how the IMS change data “maps” to the target DB2 tables.
7. Using the information from the mapping definitions, the RUP updates the target DB2 tables.
8. If the target DB2 tables are staging tables, DB2 DataPropagator can propagate the change data to almost any other relational database.

With MQ-ASYNC propagation, the DB2 tables can reflect changes to the IMS database much faster than using LOG-ASYNC propagation. If conditions are optimal, the DB2 copy can reflect the IMS updates in near-real-time.

As illustrated in Figure 1 on page 3, MQ-ASYNC propagation consists of two components and one facility:

Capture component

The Capture component is implemented in a new IMS Data Capture exit routine that continuously captures the IMS database changes and transmits them asynchronously to the Apply programs through use of the reliable messaging services of MQSeries.

The Capture component runs on the same OS/390 system as the IMS application programs that update the source IMS databases. These systems are referred to as the source systems.

Apply component

The Apply component is implemented by one or more occurrences of the IMS DataPropagator for z/OS Apply program.

The Apply component runs on the same OS/390 system as the DB2 tables that are the target of the data propagation. These systems are referred to as the target systems. Source and target systems can be on the same OS/390 image or can be different on OS/390 images.

The Apply program runs all the time and reads the MQSeries input queue associated with it. When getting an MQSeries input message containing changed IMS data, the Apply program calls the Relational Update (RUP) component of IMS DataPropagator for z/OS. The RUP maps the IMS data into relational format and applies the mapped changes to DB2 target tables.

When the source system and the target system are on different OS/390 images, the speed with which the IMS updates are propagated to DB2 depends on:

- The teleprocessing links between the two systems.
- The performance of the SQL DB2 updates to the target tables.

When the teleprocessing links between source and target system are not congested and the performance of the DB2 updates can keep up with the update rate of the propagated IMS source segments, the delay between the update of the IMS source databases and the update of the DB2 target tables can be a matter of seconds.

As explained in following sections of this chapter, it is possible to run multiple occurrences of the Apply program in parallel. When propagating a high rate of changes, this can improve the throughput and decrease the propagation delay.

Event Marker facility

The Event Marker facility (not shown in Figure 1 on page 3) enables the Apply program to stop automatically after the content of the DB2 target tables reflects a user-specifiable source system point-in-time (for example, the logical end of a business day). The IMS DataPropagator for z/OS Event Marker facility is similar to the IMS DataPropagator for z/OS Timestamp Marker facility, as delivered in previous releases of IMS DataPropagator for z/OS.

IMS-to-IMS propagation

Figure 2 shows how MQ-ASYNC propagation from IMS-to-IMS works when the IMS source and IMS target are on different z/OS images.

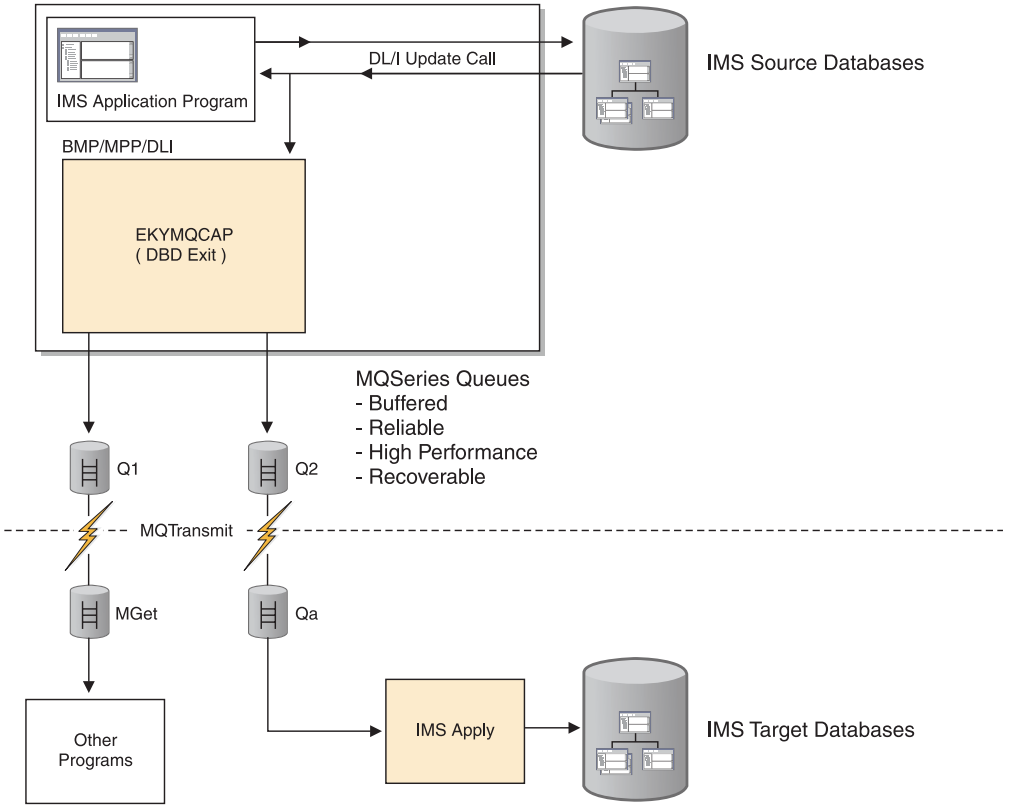


Figure 2. Propagation phase: MQ-ASYNC asynchronous IMS-to-IMS propagation

The following list describes Figure 2:

1. When an IMS application changes an IMS segment type to be propagated, DPROP captures the update using a Capture exit routine. This “capture” happens during the unit-of-work (UOW) of the IMS application. Control is passed to the Capture exit routine after the update to the segment is made.
2. The Capture exit routine builds an MQSeries message containing the information about the changed segment and places the message on an

MQSeries message queue (labeled Q2 in Figure 2 on page 5). The Capture exit routine can place the message on multiple queues (for example those labeled both Q1 and Q2), as shown in Figure 2 on page 5.

3. If all processing performs with no problems (update to the segment, building of the message, and placing the message on the MQSeries queue), the update to the segment is committed. The IMS application's UOW is now complete.
4. After the segment change is committed, MQSeries transmits the message to another MQSeries queue (labeled Qa in Figure 2 on page 5) on the system where the IMS Apply program is running. This second system is called the target system.

   The changed IMS data flows from the Capture component to the IMS Apply program occurrences as part of one or more *propagation data streams* (PRSTREAMs). All changed data contained in one PRSTREAM is transmitted by means of the same MQSeries queues and is applied in First-In First-Out (FIFO) sequence by the same occurrence of the IMS Apply program.
5. The IMS DPROP IMS Apply program retrieves the message from queue Qa and updates the IMS target database.

With MQ-ASYNC IMS-to-IMS propagation, the IMS target database can reflect changes to the IMS source database rapidly. When conditions are optimal, the IMS target copy can reflect IMS source updates in near-real-time.

As shown in Figure 2 on page 5, MQ-ASYNC propagation from IMS-to-IMS consists of Capture and IMS Apply components and Event Marker facility, as described below.

Capture component

The Capture component is implemented in an IMS Data Capture exit routine which continuously captures the IMS source database changes and transmits them asynchronously to the IMS Apply programs through the message services of MQSeries.

The Capture component runs on the same OS/390 system as the IMS application programs which update the source IMS databases. These systems are referred to as the source systems.

IMS Apply component

The IMS Apply component is implemented by one or more occurrences of the IMS DataPropagator for z/OS IMS Apply program.

The IMS Apply component runs on the same OS/390 system as the IMS databases that are the target of IMS-to-IMS propagation. These systems are referred to as the target systems. Source and target systems can be on the same OS/390 image or can be on different OS/390 images.

The IMS Apply program can run all the time in order to achieve continuous IMS-to-IMS propagation, if desired. Or, if preferred, the IMS Apply program can run up to a chosen point-in-time as required by the implementation. The IMS Apply program reads the MQSeries input queue associated with it. When receiving an MQSeries input message containing changed IMS data, the IMS Apply program updates the IMS target databases with the IMS source data changes.

When the source system and the target system are on different OS/390 images, the speed with which the IMS source updates can be propagated to the IMS targets depends on:

- The teleprocessing links between the source and target systems.
- The performance of the updates to the target databases.

When teleprocessing links between source and target systems are not congested and performance of the IMS target updates can keep pace with the update rate of the propagated IMS source segments, then the delay between the update of the IMS source databases and the update of the target databases can be a matter of seconds.

As explained in the following sections of this chapter, it is possible to run multiple occurrences of the IMS Apply program in parallel. When propagating a high rate of changes, this can improve throughput and decrease propagation delay.

Event Marker facility

The Event Marker facility (not shown in Figure 2 on page 5) enables the IMS Apply program to stop automatically after the content of the IMS target database reflects a user-specifiable source system point-in-time (for example, the logical end of a business day). The IMS DataPropagator for z/OS Event Marker facility is similar to the IMS DataPropagator for z/OS Timestamp Marker facility, as supported in previous releases of IMS DataPropagator.

Components and details of MQ-ASYNC propagation

The next few sections discuss the:

1. IMS DataPropagator for z/OS Capture component
2. Use of MQSeries for the transmission of the changed data
3. IMS DataPropagator for z/OS Apply component
4. IMS DataPropagator for z/OS Event Marker facility.

The Capture component: the MQ-ASYNC IMS Data Capture exit routine

EKYMQCAP is the IMS Data Capture exit routine provided by MQ-ASYNC. The IMS database administrator uses an EXIT=EKYMQCAP keyword of DBDGEN to identify which physical IMS databases (and optionally which segment types of these databases) should be propagated with the EKYMQCAP exit routine.

These DBDGEN specifications specify that EKYMQCAP be called by the DBMS when an IMS segment occurrence is changed.

During its initialization processing (when being called the first time within the current MVS task), EKYMQCAP reads a transmission specification file and stores the transmission specifications into internal virtual-storage resident control blocks. The transmission specification file is a PDS member consisting of simple, easy to understand, easy to provide, and easy to update specifications. To avoid changes of preexisting JCL statements, this file is by default allocated and deallocated dynamically by EKYMQCAP. You can use the ISPF editor (or an editor of your choice) to create, update, and view the simple definitions of the transmission specification file.

When being called by the IMS Data Capture facility, EKYMQCAP inspects its internal control blocks to determine whether the changed IMS segment occurrence

should be transmitted to one or more occurrences of the Apply program and through which Propagation Data Stream(s) and MQSeries output queue(s) the transmission is done.

If the changed segment is to be transmitted, EKYMQCAP builds an MQSeries message consisting of :

- A standard propagation message header and
- The information provided by the IMS Data Capture function (depending on options provided on the EXIT= keyword of the DBDGEN, this includes parent/ancestor data in addition to the changed segment occurrence and its concatenated key).

Then EKYMQCAP issues an MQPut to write the MQSeries message into the appropriate MQSeries queue.

When the IMS application finally commits the change to the database, MQSeries makes the MQSeries message available for further processing and/or transmission. Prior to the commit, MQSeries keeps the MQSeries messages in its buffered queues. If the IMS application abends or rollbacks, MQSeries discards the MQSeries messages written by the UOW that abends or rollbacks.

The commit processing for IMS database update operations and for inserted MQSeries messages is coordinated with a two-phase commit protocol. This ensures that all committed IMS changes (that must be propagated) and only committed IMS changes are transmitted to their destination.

EKYMQCAP behaves normally in error situations (for example, in IMS online environments, EKYMQCAP issues a ROLS call to rollback without loss of any IMS TM input messages).

Similar to the Capture component for synchronous propagation, EKYMQCAP has its own

- MQ-ASYNC status file (this file is, by default, allocated dynamically by IMS DPROP)
- VLF classes

EKYMQCAP does not have its own DPROP directory (MQ-ASYNC does not require and does not use DB2 for its Capture component).

The MQ-ASYNC Status Record records whether the capture activities of MQ-ASYNC have been stopped through use of the IMS DPROP SCU utility.

The VLF class (this is a PDS VLF class) supports frequent, high-performance access to:

- The MQ-ASYNC Status file to rapidly detect an emergency stop or reactivation of the capture activities of IMS DPROP.
- The Transmission Specification file to rapidly detect changes to transmission specifications.

Often, on one OS/390 system, all jobs that update the IMS source databases use the same Transmission Specification file and the same MQ-ASYNC Status file. However, when the same OS/390 system is used both for production and test purposes, you can use different MQ-ASYNC Status files and Transmission Specification files for the capture activities in different OS/390 job steps.

All job steps that use the same MQ-ASYNC Status file use the same MQ-ASYNC Capture system. When operators or administrators use the SCU EMERGENCY STOP and RESET control statements to update an IMS DPROP Status file, the Capture activities of the entire MQ-ASYNC Capture system (all job steps accessing the same MQ-ASYNC Status file) are affected.

Each IMS DPROP Capture system has its own default Transmission Specification file. Typically, but not necessarily, every IMS batch region job and IMS dependent online region of the same IMS DPROP Capture system will use the same Transmission Specification file.

Capture error behavior

If the MQSeries resources required by the Capture component are not available when application programs update the IMS Source databases, the Capture component abends so that no IMS input messages are lost.

In an emergency situation, when the propagation is less important then the execution of the IMS database update applications, you can:

- Use the SCU to emergency stop the Capture activities of an entire IMS DPROP Capture system; or
- Provide a PROP OFF control statement in the //EKYIN File of a particular job step that updates IMS source databases.

This results in data inconsistencies between IMS source databases and DB2 target tables. You must resynchronize the DB2 target tables with the IMS source.

The Transmission Specification file

This section provides examples and more detailed explanations of the Transmission Specification file. To understand the examples, you must understand the following:

- The database administrator (DBA) must use a QMANAGER control statement to define which MQSeries queue manager (which MQSeries subsystem) will be used by the Capture component to send the changed IMS data.
- The DBA must use PRSTREAM control statements to define one or multiple Propagation Data Streams. The changed IMS data contained in a Propagation Data Stream flows in a first-in first-out order from the Capture component to the Apply component.

  The DBA assigns a mnemonic name to the Propagation Data Stream, and defines through which MQSeries Output queue the changed data will be sent to the Apply component.

  On the target system, each Propagation Data Stream (and each MQSeries input queue) is processed by one, and by only one, occurrence of the Apply program. A Propagation Data Stream cannot be processed concurrently by more than one occurrence of the Apply program.
- For each Propagation Data Stream, the DBA uses database control statements to define which captured changed data (that is. the changed data of which IMS databases and, optionally, of which IMS segment types) will be sent as part of the Propagation Data Stream.

In the following examples, replace the information shown in italics with installation-specific names. Information that is not in italics must be provided as shown.

```
QMANAGER NAME= MQP1 ;

PRSTREAM NAME=STREAM1, QUEUE=MQ.DATAWAREHOUSE1 ;
DB ALL ;
```

Figure 3. Example 1 of a Transmission Specification file

Figure 3 consists of only one Propagation Data Stream (PRSTREAM) called MQPROD1. This Propagation Data Stream transmits (by means of the MQSeries Queue 'ODS1') *all* IMS data that IMS captures for the EKYMQCAP exit routine.

On the target systems, one particular Propagation Data Stream (and one particular MQSeries Input Queue) cannot be processed concurrently by more than one occurrence of the Apply program. Therefore, Figure 3 shows that all IMS changes captured for EKYMQCAP will be processed by one single occurrence of the Apply program.

```
QMANAGER NAME= MQP1 ;

PRSTREAM NAME=STREAM1, QUEUE=MQ.DATAWAREHOUSE1,
         MAXMSGSIZE=126000, COMPRESS=NO;
DB DBD=(CUSTOMER,ACCOUNT) ;
DB DBD=ADDRESS ;

PRSTREAM NAME=STREAM2, QUEUE=ODS03 ;
DB DBD=FOREX, SEG=(POSITION,HISTORY);
KEYRANGE DBD=FOREX,LKEY=.....HKEY=.....;
```

Figure 4. Example 2 of a Transmission Specification file

Figure 4 consists of two Propagation Data Streams (PRSTREAM):

- The first Propagation Data Stream transmits (by means of the MQSeries Queue MQ.DATAWAREHOUSE1) all the captured changes of the CUSTOMER, ACCOUNT and ADDRESS databases. It also explicitly specifies, with the MAXMSGSIZE parameter, the maximum MQSeries message size that the Capture component will build. Additionally, it requests, with COMPRESS=NO, that the MQSeries messages not be compressed. For performance reasons, an MQ message can contain multiple database updates of the same UOW to achieve a higher MQSeries message throughput.
- The second Propagation Data Stream transmits (by means of the MQSeries Queue ODS03) the captured changes to the segments POSITION and HISTORY of the FOREX database, but only for those database records with root key values located in the key range defined by the KEYRANGE control statement. It demonstrates the use of the optional KEYRANGE control statement. The KEYRANGE control statement supports a partitioning propagation schema, allowing the concurrent execution of multiple occurrences of the Apply programs for the same database; for example, one APPLY for each key-partition and PRSTREAM/queue. These multiple Apply program occurrences can run on the same or on different target OS/390 systems. For high database update rates, having multiple Apply programs can increase the throughput and/or reduce the propagation delay. A KEYRANGE control statement applies to a specific propagation data stream. The KEYRANGE control statement must come after the PRSTREAM control statement to which it applies and before the next PRSTREAM control statement.

On the target system(s) shown in Figure 4, it is possible to run two occurrences of the Apply program concurrently. One Apply program can process the first

PRSTREAM and the other Apply program can process the second PRSTREAM. If you prefer, it is also be possible to run one single occurrence of the Apply program that processes both PRSTREAMS.

Using multiple Propagation Data Streams allows IMS DataPropagator for z/OS to process the captured data by more than one occurrence of the Apply program. This can be useful, for example:

- When a subset of the captured data should be propagated in near-real-time mode and another subset of the data in point-in-time mode.
- When the captured data is propagated to different systems.
- When the IMS database update rate is high and it is important to increase the throughput of the Apply process by running multiple occurrences of the Apply program in parallel.

  There is no time-coordination between the processing of one occurrence of the Apply program and the processing of another occurrence of the Apply program. The activities of multiple concurrently running Apply/RUP occurrences are not synchronized and the scope of the source system UOWs across multiple Apply/RUP occurrences is not preserved.

You might find it practical to define one Propagation Data Stream for each IMS database or for each group of related databases.

Note: It is possible to transmit multiple Propagation Data Streams by means of the same MQSeries output queue. In this case, all data of all these Propagation Data Streams is transmitted and applied in a FIFO sequence.

```
QMANAGER NAME= MQPROD1 ;

PRSTREAM NAME=SANDIEGO, QUEUE=MQ.DATAWAREHOUSE1 ;
DB DBD=CUSTOMER ;
DB DBD=ACCOUNT, SEG=(ROOT,CREDIT,DEBIT) ;
DB DBD=ADDRESS, SEGEXCL=VACATION ;

PRSTREAM NAME=LOSGATOS, QUEUE=CORPORATE.HEADQUARTERS ;
DB DBD=ACCOUNT ;

PRSTREAM NAME=WEB, QUEUE=CORPORATE.WEBPORTAL         ;
DB DBD=CUSTOMER,
   SEG=(ROOT,
        ADDRESS,
        RATING) ;
DB DBD=FOREX ;
```

Figure 5. Example 3 of a Transmission Specification File

Figure 5 consists of multiple Propagation Data Streams. However, in contrast to the example shown in Figure 4 on page 10, only a subset of the IMS segment types of the ACCOUNT database and of the ADDRESS database are transmitted as part of the first Propagation Data Stream.

- For the ACCOUNT database, only those segment types (ROOT, CREDIT and DEBIT) explicitly specified on the SEG= keyword will be transmitted.
- For the ADDRESS database, all segment types, with the exception of the VACATION segment type, are transmitted as part of the first Propagation Data Stream.

In Figure 5 on page 11, the DB control statement for the CUSTOMER database in the third PRSTREAM demonstrates that one control statement can consist of multiple input records, as is usual for DPROP control statements. This DB control statement consists of four input records.

Important control statements in the Transmission Specification file

As shown in the previous examples, the most important control statements in the Transmission Specification file are:

PRSTREAM control statements

A Propagation Data Stream (PRSTREAM) control statement identifies a group of captured data to be transmitted together to its destination (where it is processed by an Apply program occurrence). On the PRSTREAM control statement the DBA:

- Assigns a short mnemonic unique name to the Propagation Data Stream (for example, group01).
- Provides the name of the MQSeries output queue (or the name of an MQSeries queue alias) used by EKYMQCAP to transmit the data. This queue must be defined to MQSeries and must be reserved for exclusive use by MQ-ASYNC.

Each PRSTREAM control statement must be followed by zero, one, or more DB control statements that identify the names of IMS databases whose data will be transmitted as part of the propagation stream.

DB control statements

For each physical IMS database whose data is transmitted as part of a Propagation Data Stream, the DBA must identify the name of the physical IMS DBD. This is done by providing one or more DB control statements.

By default, all captured IMS segment types of these physical DBDs are transmitted as part of the PRSTREAM (all IMS segment types defined in these IMS DBDs with an EXIT=EKYMQCAP keyword). As explained below, this default can optionally be overridden with SEG= and SEGEXCL= keywords.

The DBA can also provide on the DB statement either SEG= (segment include) or SEGEXCL= (segment exclude) keywords. The SEG= and SEGEXCL= keyword values identify the names of the IMS segment types to be included or excluded from this PRSTREAM.

Optionally (this has not been shown in the previous examples), the DBA can use FIELD statements to specify which fields of an IMS segment type are relevant for the propagation. During an IMS DB Replace, if the content of these fields has not changed, the Replace is not transmitted to the Apply program. In some cases, this can reduce the amount of transmitted and propagated data and can enhance the overall performance of the propagation process.

With MQ-ASYNC processing, you can propagate the same IMS segment to multiple systems and/or to multiple target tables.

- One method to propagate the same IMS change to multiple tables of the same DB2 system, is to define multiple IMS DPROP propagation requests for the same IMS segment type.
- Another method allows you to propagate the same changes to DB2 tables of different DB2 systems. This other method is illustrated in Figure 5 on page 11. With this method, the DBA provides multiple DB control statements for the same IMS physical DBD. In Figure 5 on page 11, the changes of the ACCOUNT DB

are part of two Propagation Data Streams and are therefore transmitted twice by EKYMQCAP: one time by means of the MQSeries queue named `mq.datawarehouse1` and a second time by means of the MQSeries queue named `corporate.headquarters`.

Control statements in the //EKYIN file

The Capture component of MQ-ASYNC supports IMS DPROP PROP LOAD, PROP OFF, TRACE and TRDEST control statements in the //EKYIN file.

1. When you provide a PROP LOAD control statement in the optional //EKYIN parameter data set of an IMS DB-LOAD job step, you indicate that IMS database inserts performed with the IMS processing option LOAD be transmitted. (the default is not to transmit changes).
2. By providing a PROP OFF control statement in the //EKYIN parameter data set of an updating job step, you indicate that changes performed by this updating job step should not be propagated or transmitted.

   The PROP OFF control statement turns off propagation of the IMS database updates of those job steps that run with a PROP OFF control statement. PROP-OFF is useful in test environments, for example, when the test team does not need data propagation. In production environments, using PROP OFF should be restricted, because its use results in inconsistencies between the IMS source data copy and the DB2 target data copies.

   In contrast to synchronous propagation, use of PROP OFF must not be explicitly allowed through use of a SCU ALLOWPROPOFF. This is because the SCU does not support ALLOWPROPOFF for asynchronous propagation; there is no DPROP directory on the source system to record the execution of ALLOWPROPOFF.
3. The TRACE control statement activates the IMS DPROP trace facilities and identifies what should be traced. This is similar to the current use of TRACE for synchronous propagation with RUP, but with a trace of MQSeries activities instead of DB2/SQL activities.
4. The TRDEST control statement overrides the default destination of the IMS DPROP trace output. This is similar to the current use of TRDEST for synchronous propagation with RUP.

If you use both SYNC and MQ-ASYNC:

If the same IMS batch region or the same IMS online dependent region needs to support both synchronous propagation with EKYRUP00 and MQ-ASYNC propagation with EKYMQCAP, the control statement in //EKYIN is used to control both EKYRUP00 and EKYMQCAP. Be aware of the following:

- EKYMQCAP ignores those //EKYIN control statements (for example, PROP SUSP and RESIDENT control statements) that do not apply to asynchronous propagation.
- EKYRUP00 enforces its own rules for the use of PROP OFF.

Use of MQSeries for message transmission

MQSeries provides an extremely reliable, high-performance, and high-throughput asynchronous transmission of messages from the Capture component to the Apply component.

The following list identifies some reasons why MQSeries provides strong asynchronous communication functions to MQ-ASYNC propagation:

- When defining messages as *persistent*, MQSeries ensures that the messages are not lost and that they are not delivered multiple times. Through the use of an efficient logging component, MQSeries provides a combination of message integrity and high performance.
- Both the MQSeries queues and the MQSeries logs are buffered efficiently. This substantially shields the updating applications from performance impact of the MQSeries activities.
- MQSeries allows the following to operate asynchronously to each other:
  - Sender (the IMS Data Capture exit routine)
  - Transmission over the TP links
  - And Receiver (this is the MQ-ASYNC Apply program)

  Therefore, each one of these components can run at its highest speed without its performance being impacted by the activities of the two other components. On large well-tuned OS/390 systems, MQSeries supports a near-real-time message delivery of around 1000 persistent messages per second. Note that such high message rates result in substantial CPU usage by MQSeries.
- Because the transmission of the messages are asynchronous to the updating IMS application programs, the IMS applications are not affected if the target system is not available or if the transmission links to the target system are not active.
- For the transmission from source to target system, MQSeries provides logic that makes efficient use of the available teleprocessing bandwidth. This also contributes to the performance and throughput characteristics.
- MQSeries supports the definition of multiple, very large message queue data sets (4 gigabytes per queue data set). These large queue data sets allow the source and target systems to operate independently from each other. The large message queue data sets also shield the Capture component and updating applications from long temporary unavailability of either the target system, the Apply program, or the teleprocessing links.
- EKYMQCAP makes sure that MQSeries transmits only committed messages and, therefore, only committed IMS changes. MQSeries waits until the updating IMS application program commits before initiating transmission of the updates of the updating program to the target system. The Apply component never receives uncommitted IMS changes.

Note the following about the use of MQSeries by MQ-ASYNC processing:

- The MQSeries message queues used by MQ-ASYNC must be dedicated to the use of MQ-ASYNC. These message queues must not be used by other MQSeries-based software.
- Because the Capture component of MQ-ASYNC uses the same MQSeries priority when sending changed IMS data, the changed IMS data of a Propagation Data Stream is delivered in a sequence that ensures that the updates are applied to the target tables in the correct sequence. The messages of a Propagation Data Stream are delivered in FIFO sequence to the Apply component.

  Note: There is one exception to FIFO message delivery sequence. The following illustrates this exception:
  1. UOW A is writing **Message A** into an MQSeries queue.
  2. Later, UOW B is writing **Message B** into the same MQSeries queue.
  3. UOW B commits before UOW A.

  In this case, if MQSeries is ready to transmit the committed messages of UOW B before UOW A reaches its commit point, then it is possible that MSG B gets delivered before MSG A. Because of the way that the DBMSs enqueue modified

data, this minor exception to the FIFO rule is not significant for propagation process and for its users (the exception being the STOPAT control statement, if the Apply process is not stopped at a quiesce point).

- The input queue of the Apply program must not be read concurrently by more than one Apply program occurrence. This, too, is to ensure that updates are applied to the target tables in the correct sequence.
- Do not define the MQSeries queues so that on the target system, the input queue of an Apply program is supplied messages by multiple source system MQSeries queues.
- The MQSeries system used by MQ-ASYNC must be defined without a Dead Letter Queue. This is to avoid the exceptional situations where some messages might get stored in a Dead Letter Queue and eventually be processed in the wrong sequence, that is, a sequence other than FIFO.
- You should consider using a dedicated MQSeries queue manager for MQ-ASYNC propagation. Use of a dedicated MQSeries manager can have the following advantages:
  1. If short propagation delays are important.

     The use of a dedicated MQSeries queue manager can shield propagation activities from the MQSeries activities of other applications.
  2. If the data propagation is mission critical.

     The use of a dedicated MQSeries manager can shield propagation activities from other MQSeries activities that have not yet been tested or implemented.
  3. If you are propagating a very large number of IMS updates with MQ-ASYNC

     The use of a dedicated MQSeries manager can shield the MQSeries activities of other applications from the high-use of MQSeries by MQ-ASYNC.
- When defining the MQ queues used by IMS DPROP, it is important that you define the queues:
  1. With the DEFPSIST(YES) attribute to avoid the loss of MQ messages due to a restart of the queue manager.
  2. With a sufficiently large MAXDEPTH value to support a sufficiently large number of messages on the queues. The page sets used for the queues should also be sufficiently large. The number of messages might be large if some resources are not available. The resources include the target system, the link between source and target systems, the Apply programs, and a target database.

The Apply program

The Apply program occurrences are started like other batch jobs or tasks; typically, by standard automated operation tools.

The Apply program occurrences can be stopped:

- With an OS/390 operator `STOP jobname` command
- When reading an MQSeries message whose source system timestamp is greater than a specifiable timestamp
- When reading an Event Marker MQSeries message identifying a source system point-in-time, at which the Apply stops. As explained later in this chapter, this option is often preferable to the previous option of stopping at an explicitly specified timestamp.

When used for a near-real-time propagation, the Apply program occurrences are permanently active.

When used for a point-in-time propagation, the Apply program occurrences are started by the Operator when the target tables must be updated. They can stop automatically when processing the Event Marker STOP APPLY MQSeries message identifying the source system point-in-time that must be reflected in the content of the target tables.

You can run either a single or multiple occurrences of the Apply program. Running multiple Apply program occurrences allows an increase of throughput. Note that a particular MQSeries input queue and a particular PRSTREAM cannot be concurrently processed by more than one Apply program occurrence, and that one occurrence of the APPLY cannot process more than one MQSeries input queue.

An Apply program uses a control statement file occurrence to understand which MQSeries queue it should use as input. This MQSeries queue is opened for exclusive use by the Apply program occurrence and is permanently polled for a message input.

Each MQSeries message contains one IMS source change. An MQSeries message can also contain multiple IMS source changes. After retrieving a message from an MQSeries input queue, with IMS-to-DB2 propagation the Apply program calls the RUP component as the RUP component is called for synchronous propagation by the IMS Data Capture function. The message sent by EKYMQCAP contains all data required for the RUP call.

The RUP uses the Propagation Request (PR) mapping definitions of the IMS DPROP directory to understand how to map the changed IMS data into relational format. It then issues SQL statements that are required to apply the mapped change to the target DB2 table.

By default, RUP will process *all* PRs that propagate the changed IMS segment type. The DBA can optionally identify a *subset* of PRs to be processed. This can be convenient, for example, when the same IMS DPROP directory contains multiple PRs for the same source IMS data, two for MQ-ASYNC propagation and the others for LOG-ASYNC propagation. Both MQ-ASYNC PR definitions are processed in near-real-time mode; the LOG-ASYNC PRs are processed in point-in-time mode.

Error behavior

With IMS-to-DB2 propagation, the Apply program checks the RUP's return codes and provides the logic to handle any propagation failure.

Typically, the DBA can use the default options of the error handling logic of the Apply program. For special situations, these defaults can be overridden by user-provided control statements.

With the default options, the error handling logic of the Apply program can be summarized as follows:

- For deadlocks and time-outs, the Apply program retries the processing of the deadlocked or timed out UOW without loss of any data and without abending. This is possible because it issues an INIT STATUS GROUPB call.
- For other errors, the Apply program writes error messages and traces and then abends if the PR has not been defined with ERROPT=IGNORE[1]. Based on the

1. Only applicable with IMS-to-DB2 propagation.

description of the error messages, you can fix the problem and restart the Apply program. This will not result in any loss of data.

For PRs that are defined with ERROPT=IGNORE[1], if the error belongs to the category of mapping errors, the Apply program will:

- Write error messages.
- Write the database change that caused the problem in an error table.
- Skip further processing of the propagation of the database changes that resulted in problems.

Defining PRs with ERROPT=IGNORE[1] can be useful, for example, when you are starting to propagate new data and are not yet sure if your propagation definitions are correct or if the source IMS data is very clean (for example, when an IMS source field that is supposed to be numeric contains nonnumeric data).

The error option ERROPT=IGNORE[1] is specified when the PR is created with MVG and can be changed by the SCU utility. ERROPT=IGNORE[1] applies to the level of the individual PR. It is useful when initial problems with specific PRs are expected. Other Apply control statement FAILURES, which apply to an entire Apply job step and to all PRs processed in that Apply job step, can be used to control the logic of the Apply program in handling propagation failures, like replacing a nonexisting row.

By default, when FAILURES control statements are not provided, the Apply program abends without losing any MQSeries input message when encountering a propagation failure. Before abending, the Apply program documents the problem with:

- WTOs
- Problem descriptions on //EKYPRINT
- Audit/SMF records
- Trace information

With FAILURES control statements, you can specify:

- Which category of failures should not result in an abend. For these types of failure, the Apply:

  Records the IMS change that could not be applied:

  - for IMS-to-IMS replication - in an error database
  - for IMS-to-DB2 replication - in an error table
- Documents the failure.
- Continues processing with the next MQSeries input message.
- For IMS-to-IMS and IMS-to-DB2 propagation, the following failures categories are applicable as CATEGORY parameters of the FAILURES control statement:
  - DATA (appropriate for IMS-to-IMS propagation only) for example, for failures of the type where the IMS Apply program is processing the Replace of a segment occurrence that does not exist in the target database.
  - MAPPING (appropriate for IMS-to-DB2 propagation only) for PR mapping-related failures, such as nonnumeric data in a field intended to be numeric, or a nonexisting row to be replaced or deleted.
  - UNAVAILABLE for availability-related failures such as when IMS databases are not available.
  - MISC for miscellaneous failures such as reading an MQSeries message that does not have a correct IMS DPROP MQ-ASYNC format.

- The maximum number of failures that an Apply program can accept before abending. This can be useful because it might be acceptable to allow a few failures but unacceptable to continue processing if thousands of database records will be affected.
- The maximum number of failures that the Apply program can document. This can be used to avoid flooding consoles, printing files, and the error table or error database with thousands of messages or records.

Commit processing

The Commit processing of the Apply program does not preserve the scope of the IMS UOWs.

This means that two different IMS updates of the same IMS UOW can be applied within two different DB2 UOWs.

Note that multiple occurrences of the Apply program do not coordinate their processing. Therefore, when two different Apply program occurrences propagate changes of the same IMS UOW, the application of these two Apply occurrences are not done within the same DB2 UOW.

Staging tables

Section “Setting the technical columns of the staging tables” on page 178 describes how MQ-ASYNC sets the values of various technical columns of the staging tables.

Performance statistics

If you experience performance problems, MQ-ASYNC includes performance statistics, providing information on both throughput and propagation delays (the propagation delay is the elapsed time between the source IMS database update and its application to the target DB2 table). These statistics are written to SMF and can be retrieved with the IMS DPROP Audit Extract utility. This utility loads the SMF records into a DB2 table; you can then use QMF or another query tool to create performance reports.

Through the use of control statements in the //APPLYIN file, you can specify the detail level of the performance records written to SMF. The control statements can also specify at which frequency the performance records are written to SMF.

The statistics about propagation delays are useful only if the clocks of the source system and target system have been sufficiently synchronized (plus or minus 1 second). This can be done through exploitation of the OS/390 ETR capabilities.

MQSeries also provides performance related statistics. MQSeries provides a TOTLATNT and VALIDGET count for each queue. By dividing the TOTLATNT value by the VALIDGET value, you can determine the average time that an MQSeries message was in that particular queue.

Other statistics

When ending normally, the Apply program also writes statistics to a print file that shows how many MQSeries messages it has processed, how many times the RUP has been called, and how many times various types of error conditions have occurred.

The Apply input data set used for IMS-to-IMS propagation

Each occurrence of an IMS DPROP MQ-ASYNC IMS Apply program has an Apply input data set as input. This can be a DD * file or a member of a PDS. This section provides an overview of the control statements of this parameter file.

The most important control statements in the Apply input data set for the IMS Apply program are:

- The **QUEUE** control statements: The database administrator must provide a QUEUE control statement to identify the MQSeries input queue which the IMS Apply program processes as input.
- Optional **STOPAT** control statements: The STOPAT control statement identifies either a source system timestamp or a source system event marker. In the first case, the IMS Apply program stops when it encounters MQSeries messages with a source system timestamp higher than the specified timestamp. In the second case, the IMS Apply program stops when reading the event marker message containing the specified event marker ID.
- Optional **FAILURES** control statement which is used to override the default error handling options of the IMS Apply program. By default, the IMS Apply program backs out and abends when it encounters errors. However, by providing FAILURES CATEGORY=*xxxxxx* ACCEPT=*nnnnn* control statements, the IMS Apply program abends only after encountering *nnnnn* errors of the specified error category.

  The FAILURES control statement can be used to accept or tolerate a specifiable number of errors. Using FAILURES ACCEPT=*nnnnn* results in IMS changes being discarded. These IMS changes are stored by the IMS Apply program in the IMS MQ-ASYNC error database.

The IMS Apply input data set has the following additional control statements:

- A control statement to specify the name of the IMS Apply occurrence (each IMS Apply occurrence has a unique name).
- A control statement to specify the name of the MQSeries queue manager (the name of the MQSeries subsystem).
- An optional control statement for tuning purposes to override the default commit frequency of the IMS Apply.
- An optional control statement to override the default options for the periodic writing of performance statistics.
- Optional control statements to specify that the IMS Apply program should verify that the "before-image" of a segment sent by the IMS source system matches the segment data in the target database segment which is to be replaced or deleted.
- Optional control statements to identify segment types that do not have a unique key but do have multiple occurrences under their parent.

The Apply input data set used for IMS-to-DB2 propagation

Like the IMS DPROP Receiver, each occurrence of an Apply program has as input an Apply input data set. This can be a DD * file or a member of a PDS. This section provides an overview of the control statements of this parameter file.

The most important control statements in the Apply input data set are:

- The **QUEUE** control statements.

  The DBA must provide a QUEUE control statement to identify the MQSeries input queue that the Apply program processes as input.

- Optional **STOPAT** control statements.

  The STOPAT control statement is used to automatically stop the Apply program.

  The STOPAT control statement identifies either a source system timestamp or a source system event marker. In the first case, the Apply program stops when it encounters MQSeries messages with a source system timestamp higher than the specified timestamp. In the second case, the Apply program stops when reading the event marker message containing the specified event markerId.

- Optional **INCLUDE** and **EXCLUDE** control statements.

  By default, an APPLY program occurrence (more precisely, the RUP occurrence called by the Apply program) processes all PRs that propagate the changed IMS segments contained in the MQSeries message.

  The DBA can override this default by providing one or multiple INCLUDE control statements that identify explicitly which PRs to include in the Apply processing. The INCLUDE control statements can identify these PRs by using one of the following keywords:

  PR=
  PRSET=
  DBD=
  A pair of DBD= and SEG=

  The DBA can also use EXCLUDE control statements that identify explicitly which PRs to exclude from the APPLY processing.

  Optionally, the EXCLUDE control statement can be provided with an UNTIL keyword, specifying that the exclusion applies only to IMS changes that occurred until the specified source system timestamp. This can be useful in those rare cases where the source and target data copies become inconsistent and you decide to resynchronize source and target copies with an Extract/Load process. In this situation the MQSeries input queue of the Apply might contain old IMS changes (changes that are older than the new extract) that should not be applied or processed by the Extract process. In this case, specify the source system timestamp of the Extract process on the UNTIL= keyword.

- Optional **FAILURES** control statement used to override the default error handling options of the Apply program.

  By default, the Apply program backs out and abends when the RUP encounters errors other than mapping errors of PRs defined with ERROPT=IGNORE. However, by providing FAILURES CATEGORY=*xxxxxx* ACCEPT=*nnnn* control statements, the Apply program abends only after encountering *nnnn* errors of the specified error category.

  The FAILURES control statement can be used to accept or tolerate a specifiable number of errors.

  Some differences in the uses of ERROPT=IGNORE and FAILURES ACCEPT=*nnnn* are as follows:

  – ERROPT=IGNORE is at the PR level and is useful for individual PRs when new PR definitions have not yet been thoroughly tested or when you know that due to questionable source data, particular PRs will probably encounter problems that should not affect the processing of other PRs.

  – FAILURES ACCEPT=*nnnn* is useful in emergency situations after encountering unexpected errors. Use of FAILURES ACCEPT=*nnnn* allows IMS DataPropagator for z/OS to keep running when there are a limited number of unexpected errors. These errors might not necessarily be associated with a particular PR; for example, after a cold start of MQSeries or of an MQSeries link. The Apply program can then attempt to replace a row that has never been inserted, because the insert was lost as part of the cold start.

Using FAILURES ACCEPT=*nnnnn* results in IMS changes that are discarded. These IMS changes are stored by the Apply program in a special DB2 table, the MQ-ASYNC error table.

The Apply input data set is similar to the Receiver input data set and has the following additional control statements:

- A control statement to specify the name of the Apply occurrence. Each Apply occurrence has its own unique name.
- Control statements to specify the name of the DB2 subsystem and the name of the DB2 plan.
- A control statement to specify the name of the MQSeries queue manager (the name of the MQSeries subsystem).
- An optional control statement used for tuning purposes to override the default commit frequency of the Apply.
- Optional control statements to control the detail level and frequency of performance records written to SMF.

The MQ-ASYNC Event Marker facility

The MQ-ASYNC Event Marker facility simplifies the coordination of propagation-related activities across system boundaries and across systems that operate asynchronously to each other. The MQ-ASYNC Event Marker facility is similar to the current IMS DataPropagator for z/OS timestamp marker support[2].

The Event Marker facility runs on the IMS source system to create MQSeries messages that tell the Apply program the source system point-in-time at which an event (for example, the start of a database extract, the logical end of a business day, or a database quiesce point) happened on the source system.

The MQ-ASYNC event marker facility consists of two parts:

1. On the source system, a user runs the event marker facility to identify and record the occurrence of an event and to send an event marker MQSeries message to the Apply program(s). The event marker facility must be run when the identified event occurs. The user is typically a DBA or operations specialist, but can be automated operations tools.

   The user assigns an ID to the event and specifies by which Propagation Data Stream(s) (PRSTREAMs) the MQSeries message will be transmitted to the Apply program(s).

   The MQSeries message containing the event marker will be transmitted with all captured IMS changes in FIFO sequence to the Apply programs.
2. On the target system, when retrieving an MQSeries message containing an event marker the Apply program will record the event marker into a special DB2 table and inform the operators that the message with the event marker was received.

   Then, if one or more STOPAFTER control statements of the Apply specifies that the Apply stop when reading an event marker with a specifiable ID, the Apply program checks whether the event marker ID contained in the MQSeries message matches one of the IDs specified on the STOPAFTER control statements.

2. Timestamp marker support controls both the Selector and the Receiver for LOG-ASYNC propagation. The Event Marker facility controls only the Apply program of MQ-ASYNC propagation. The Capture component of MQ-ASYNC always operates in real time and does not need timestamp marker support.

- If there is no match, the Apply program will continue its processing by reading the next MQSeries Message.
- If there *is* a match, the Apply program stops its processing. Since the event marker messages are transmitted in FIFO sequence in the same message stream as the messages containing the IMS data changes, the processing sequence of all these messages by the Apply program on the target system, and the sequence of their creation on the source system are the same. This is important. This means that the event marker can stop the Apply program on the target system when the content of the target copies reflects the source system point-in-time when the event marker utility was invoked on the source system.

The Event Marker can be also used to test whether the MQSeries transmission of messages is functioning properly from end-to-end because the Apply program will issue a WTO when processing the event marker.

On the source system, the event marker facility can be executed as part of the IMS DPROP Capture System utility (CUT) or it can be called by application programs.

Restrictions of MQ-ASYNC Propagation

The following restrictions apply to IMS DataPropagator for z/OS V3.1 MQ-ASYNC propagation:

1. Currently, MQ-ASYNC does not support propagation of IMS database changes performed in environments where IMS does not support the IMS Changed Data Capture Function. With IMS DataPropagator for z/OS V3.1, IMS Changed Data Capture is not supported in CICS or ODBA environments. MQ-ASYNC does not support propagation of IMS database updates made from these environments.
2. IMS DataPropagator for z/OS V3.1 MQ-ASYNC propagation will not preserve the scope of the UOWs of the source system on the target system. This means that if an IMS application program of the source system performs two IMS database updates within the same IMS UOW, these two updates are often applied by the MQ-ASYNC Apply program in two different DB2 UOWs.
3. IMS DataPropagator for z/OS V3.1 MQ-ASYNC does not support the ROLS calls with tokens.
4. If the MQ system is running with MQSeries shared queues, you can update IMS databases from more than one OS/390 image, when using the shared queue.

Chapter 2. Administrative tasks for MQ-ASYNC

This chapter summarizes the administrative tasks you perform for MQ-ASYNC propagation. The tasks are divided into phases:

- Mapping and design—determining what data you want to propagate
- Setup—ensuring that IMS, DB2, IMS DataPropagator for z/OS and MQSeries are ready for propagation
- Propagation—beginning and refining propagation
- Maintenance and control—periodically checking for data consistency or adjusting propagation requests

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 required. You might also repeat tasks in various phases of propagation. For example, you extract and load data during setup phase but you can also extract and load data during the maintenance and control phase in order to synchronize your data.

We recommend that you read and understand *IMS DataPropagator for z/OS: Concepts*, SC27-1544 before reading further in this chapter.

Table 1. Summary of Task Steps for MQ-ASYNC Propagation

1. Install IMS DPROP. Refer to *IMS DPROP Installation Guide* for details.
Mapping and Design Phase (with IMS-to-DB2 propagation)
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. • Determine the mapping case or design your own mapping. See Part 2, “Mapping and design,” on page 27.
3. Clean up your 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 97. Also see “Supported field formats and conversions ” on page 80 for specifics on field specifications and “Propagating variable-length segments (IMS-to-DB2) ” on page 61 for specific rules and limitations.
4. Create exit routines, if needed. Exits must be coded and compiled into the IMS DPROP library. See “Exit Routines” in Chapter 1 of *IMS DataPropagator for z/OS: Concepts*, SC27-1544 for a general description of when to use exit routines, and refer to the *IMS DPROP Customization Guide*, SC27-1214 for details on creating exit routines.
Setup Phase
5. Set up application programs involved in MQ-ASYNC Capture. See “Application programs involved in MQ-ASYNC Capture” on page 103 for further information.
6. Set up JCL of Job step involved in MQ-ASYNC Capture. See “JCL of job steps involved in MQ-ASYNC Capture” on page 104 for further information.

Table 1. Summary of Task Steps for MQ-ASYNC Propagation (continued)

7. Set up the number of Apply program occurrences, the number of MQSeries queues, the definition of PRSTREAMs in EKYTRANS file. See “Apply programs, MQSeries queues, and PRSTREAM definitions” on page 104 for further information.
8. Set up MQ-related activities, see “MQSeries-related setup activities” on page 105 for further information.
9. In IMS, (create or) modify the database definition (DBD) of each database to specify EXIT=EKYMQCAP for the segments to be propagated. See “Creating or changing DBDs ” on page 107. With IMS-to-IMS propagation, you should normally specify EXIT=EKYMQCAP at the DBD level in order to identify the entire IMS source database for propagation. With IMS-to-DB2 propagation, you are identifying segments which are subject to propagation and providing MVGIN segment structures. Run DBDGEN after creating or modifying the DBDs. You must also run ACBGEN if the database is referred to in an online IMS environment.
10. With IMS-to-IMS propagation, if the propagation target IMS databases do not exist, create these IMS target databases. With IMS-to-DB2 propagation, if the propagation target DB2 databases do not exist, create these DB2 target databases. If the propagated DB2 tables don’t already 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 112.
11. With IMS-to-DB2 propagation, use the DataRefresher MCE or the IMS DPROP MVGU to: • Populate MVG tables with propagation request definitions. • Build propagation requests. See Chapter 8, “Defining and changing propagation requests ,” on page 115.
12. If you are not using the DB2 package bind function, in DB2 bind the DB2 plans with the new or changed DBRMs. (This step is not necessary if you use the package bind function.) You are binding the RUP plans. See Chapter 10, “Binding and administering plans ,” on page 139, for information on how to bind.
13. In IMS, establish a synchronization point from which to start propagation. See “Establishing the start conditions for IMS DPROP Asynchronous MQSeries propagation ” on page 113 for procedure steps and see “Synchronization scenario with an Extract/Load” on page 183 for synchronization suggestions and methods.
14. Extract and load data. 1. For IMS-to-IMS propagation, use your desired method to populate target IMS databases, such as HD unload and HD reload utilities. During this activity, you are creating complete IMS target copies of the IMS source databases. 2. For IMS-to-DB2 propagation, use SCU and DataRefresher or your own programs. During this activity, you are creating DB2 tables that are copies of IMS databases. See Chapter 12, “Extracting and loading data ,” on page 149.
Propagation Phase (Regularly Scheduled Activity)
15. Run IMS application programs which update the databases. This will create changed data messages to be placed on the MQSeries queues.
16. If the source IMS and target IMS and DB2 systems are on different MVS images, MQSeries Transmission facilities are used to transport messages from the MQSeries queue used by the source IMS system to an MQSeries queue on the target IMS or DB2 MVS image. See “Remote site considerations ” on page 181.

Table 1. Summary of Task Steps for MQ-ASYNC Propagation (continued)

17. In IMS DPROP: 1. For IMS-to-IMS propagation, run the IMS Apply program to retrieve IMS source database changed data messages from the MQSeries queue and update the IMS target databases. 2. For IMS-to-DB2 propagation, run the Apply program to retrieve IMS source database changed data messages from the MQSeries queue and pass them to the RUP, which uses them to update the DB2 target tables.	run the See “Using the Apply program for IMS to DB2 propagation” on page 168.
Maintenance and Control Phase (Periodic Activity with IMS-to-DB2 propagation)	
18. Optional: Use the CCU to compare IMS and DB2 data and check for consistency.	See Chapter 16, “Verifying consistency between IMS and DB2 data copies (CCU),” on page 187, to understand how you can use CCU and “CCU considerations for MQ-ASYNC propagation ” on page 189. Also refer to the *Reference*, SC27-1210 for detailed information about the CCU.
19. Delete, replace, and rebuild propagation requests, as needed. See information on deleting, replacing, rebuilding, and revalidating propagation requests beginning on page 128.	
20. Revalidate propagation requests, especially if changes have been made to either the IMS DBDs or DB2 tables.	See “Revalidating propagation requests ” on page 129. You use MVGU to run a REVALIDATE function. The MVGU is described in detail in the *Reference*, SC27-1210.

Part 2. Mapping and design

These topics cover the mapping and definition phase of MQSeries-based data propagation:

- Chapter 3, “Decisions affecting mapping and propagation ,” on page 31 provides information to help you make decisions on the design of your propagation environment. Major topics included in this chapter are:
  - defining propagation requests
  - using mapping cases
  - using options with generalized mapping cases, such as IMS DPROP tables and files

  In addition, this chapter provides operational environment scenarios that help you understand and prepare for an MQSeries-based propagation environment.
- Chapter 4, “Propagation guidelines, rules, and restrictions ,” on page 59 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 completed and rules should be considered.
- Chapter 5, “Control information and environment,” on page 89 presents:
  - IMS DPROP control information
  - IMS DPROP global master timestamp
  - 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.

Chapter 3. Decisions affecting mapping and propagation

Note: Although this book is primarily concerned with MQ-ASYNC propagation, LOG-ASYNC and synchronous propagation are mentioned in this chapter for the purpose of comparison.

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 chapter guides you through the process by providing information on designing propagation. The topics described are:
- Propagation requests, supported propagation directions, and the different 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 *IMS DataPropagator for z/OS Reference*, SC27-1210 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 8, “Defining and changing propagation requests ,” on page 115 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 type of DPROP propagation being used. For MQ-ASYNC you can only specify hierarchical-to-relational propagation, that is, with MAPDIR=HR.

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[3]

One exception to this rule is if you want to propagate the same segment with TW propagation requests and additional HR propagation requests. Additional HR propagation requests propagate the segment to tables other than those for the TW propagation requests. Also consider for HR propagation requests:

- They 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 IMS DPROP or user asynchronous propagation with the IMS Asynchronous MQSeries 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 the 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 only IMS-to-DB2 for MQ-ASYNC.
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

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

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 section gives more detail on the characteristics and requirements of PRTYPE=E, summarized in Table 2 on page 33.

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 47 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. To do a DB2 extract and IMS load of propagated data for synchronous propagation, you can use the IMS DPROP DLU. The IMS DPROP mapping done during synchronous propagation is compatible with the mapping done by DLU during the DB2 extract and IMS load of data. (synchronous)
5. 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.
6. A strict set of rules for the mapping of keys exists. See “Key mapping rules by propagation request type ” on page 66.
7. IMS DPROP checks that DB2 RIRs are compatible with IMS parent/child relationships.

   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.
8. 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.
9. 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.
10. 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.
11. An IMS field (or part of one) mapped to the DB2 primary key cannot be mapped to more than one column.
12. 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.
13. All IMS fields in an entity segment that are included in a WHERE clause must be mapped to the DB2 table.
14. 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.
15. 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 section gives more detail on the characteristics and requirements of PRTYPE=L, summarized in Table 2 on page 33.

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 47 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 66.
7. IMS DPROP checks that DB2 RIRs are compatible with IMS parent/child relationships.

   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 33.

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
  – DLU extract and load activities **synchronous**
- 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 1.2 and following releases handle PRTYPE=F and PRTYPE=L the same way.

In IMS DPROP 1.1, you could define PRTYPE=F, L, and U. The CCU supported PRTYPE=F but not PRTYPE=L.

In IMS DPROP 1.2 and IMS DPROP 2.1, CCU supports both PRTYPE=L and PRTYPE=F, with no distinction. For compatibility with IMS DPROP 1.1, you can still define PRTYPE=F in IMS DPROP 1.2 and 2.1. 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 IMS DPROP 1.1 rules.

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

- Install IMS DPROP 2.1 first. Do not change your existing PRTYPE=F until you are sure migration to IMS DPROP 2.1 is successful.
- If you need any of the functions provided by the more powerful PRTYPE=E, 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.
- 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 book 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 the generalized mapping. Refer to the *IMS DataPropagator for z/OS Customization Guide*, SC27-1214 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 it 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 6 on page 39, 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.

If propagation is from DB2 to IMS, the arrows in Figure 6 on page 39 would simply be reversed.

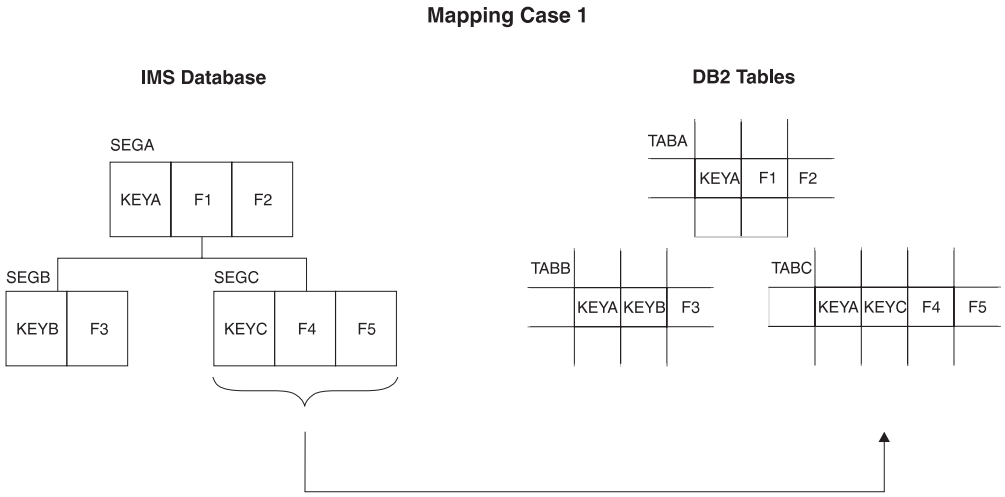


Figure 6. Mapping Case 1

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 47 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 7 on page 40, 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 7 on page 40, 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 7 would simply be reversed.

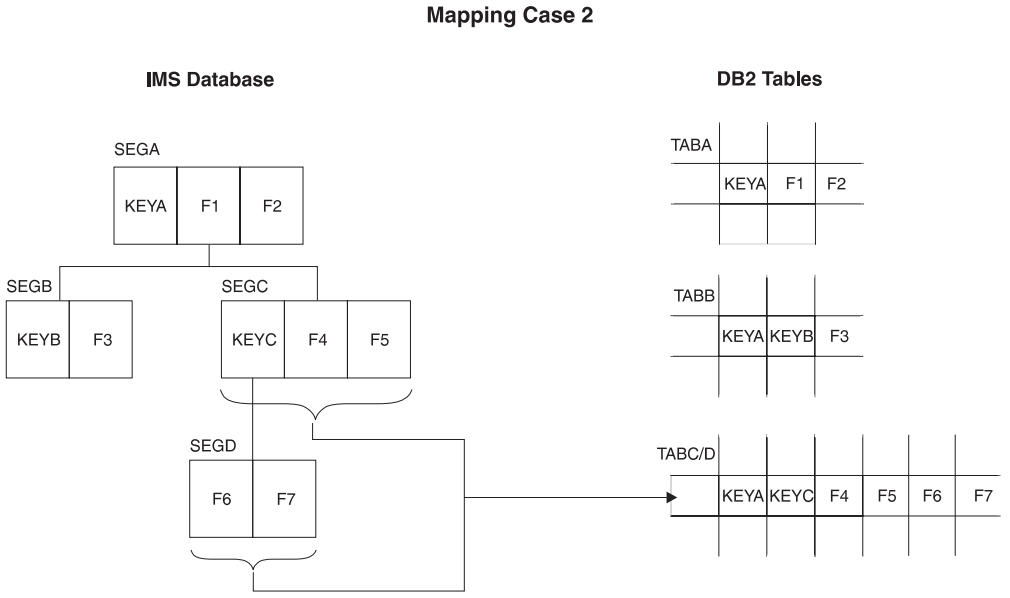


Figure 7. Mapping Case 2

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 47 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.

The following sections discuss mapping case 2 in association with:

- Rules for mapping fields in extension segments
- Extension segment for DB2-to-IMS synchronous propagation and PRTYPE=E
- DB2-to-IMS synchronous propagation to extension segments

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 8 on page 42, 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 8 on page 42, IMS segment SEGB contains two embedded structures, C and D. C and D occur multiple times within SEGB.

Figure 8 on page 42 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.

If propagation is from DB2 to IMS, the arrows in Figure 8 on page 42 would simply be reversed.

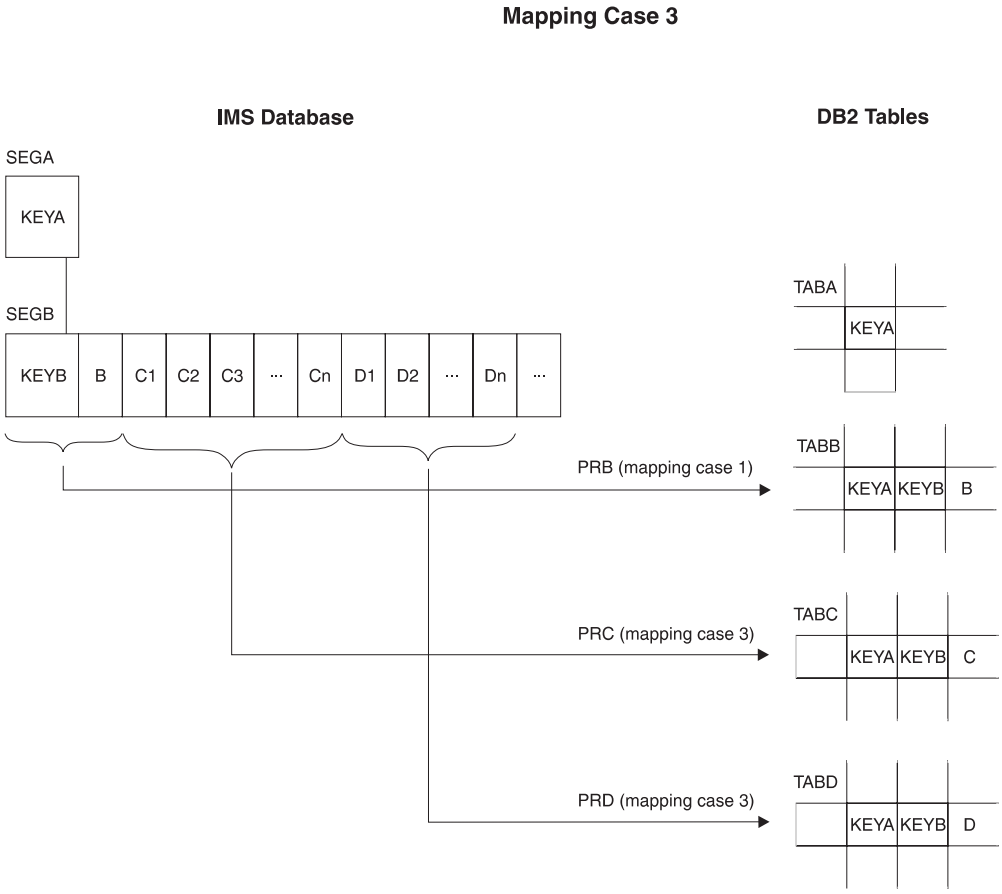


Figure 8. Mapping Case 3

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 47.

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 9 on page 43.

- 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

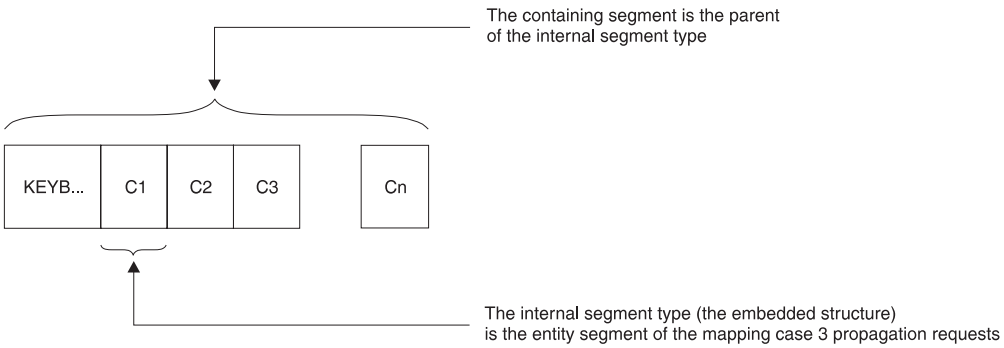


Figure 9. Containing segment and internal segment type

IMS DPROP supports both fixed- and variable-length internal segments. The *Reference*, SC27-1210 describes how you specify the fixed or variable length of internal segments to IMS DPROP.

IMS DPROP's 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 64.
- 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 10 on page 44.

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*.

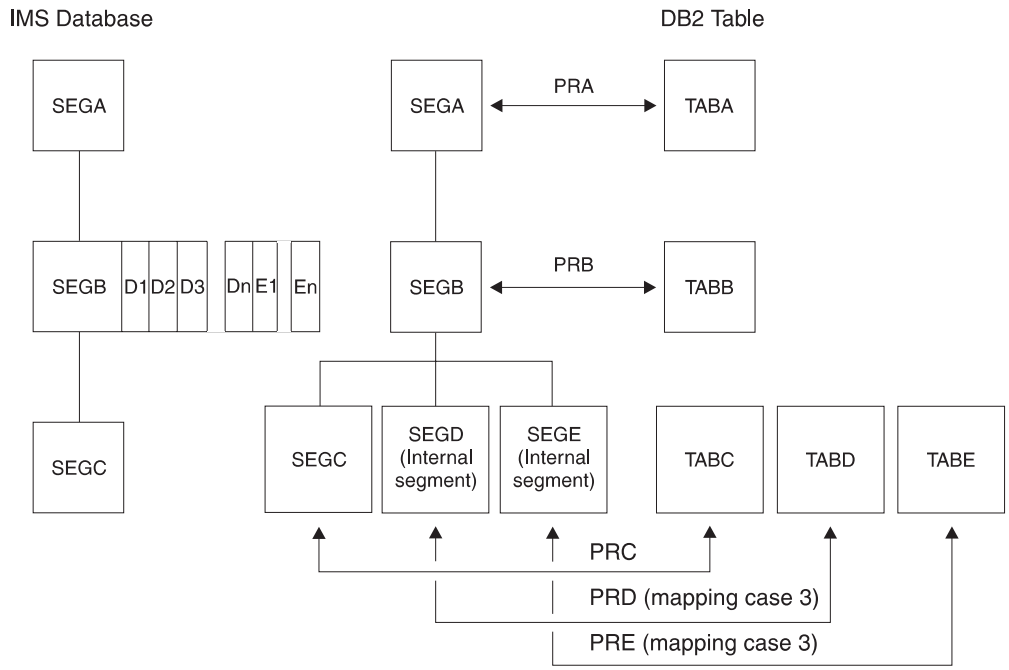


Figure 10. Conceptually normalizing the database for mapping case 3

Figure 10 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 10, 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 need 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 many advantages over user mapping, such as:

- Support for CCU, DataRefresher and DLU. (synchronous)
- Segment exit routines are easier to write than Propagation exit routines. You do not need to provide SQL logic. When doing DB2-to-IMS synchronous propagation, you also do not need to provide DL/I logic.
- You can use IMS DPROP's 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 66 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.

DB2-to-IMS Your Segment exit routine is called both for a DB2 change to the target table of the containing segment and for a DB2 change to the target of the internal segments. (synchronous)

Refer to the *IMS DataPropagator for z/OS Customization Guide*, SC27-1214 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*, SC27-1214 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 11 on page 48 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 11 on page 48.

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.

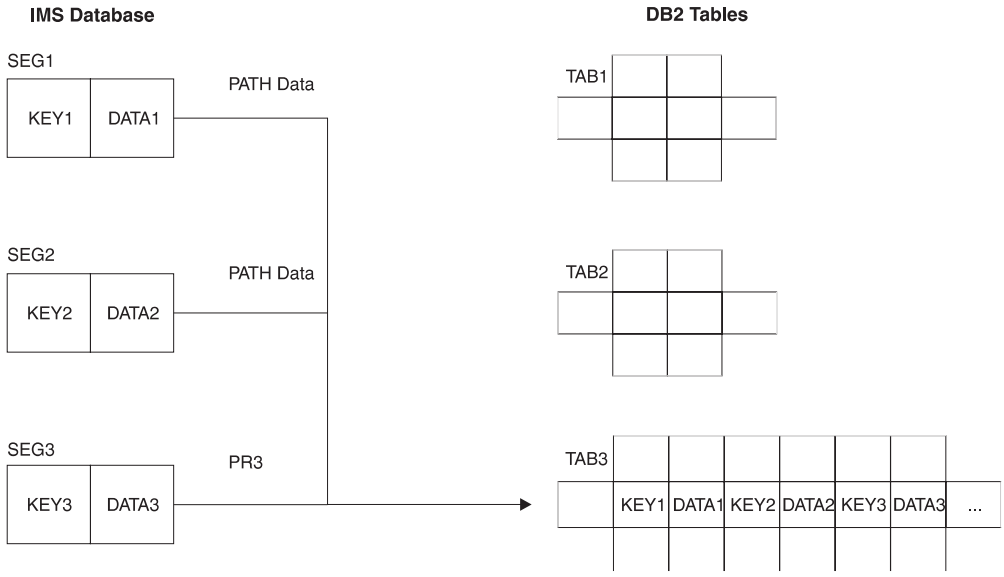


Figure 11. Mapping case 1 propagation request propagating PATH data

Uses of PATH data

IMS DPROP supports two different uses of PATH data. Because the IMS DPROP 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 88). 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 12 on page 50 assumes 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 not, IMS DPROP does not issue SQL statements to propagate the PATH data to the target table.

Figure 12 on page 50 shows 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 12 on page 50, including PATH data denormalizes the DB2 copy of the data in order to improve performance of DB2 queries.

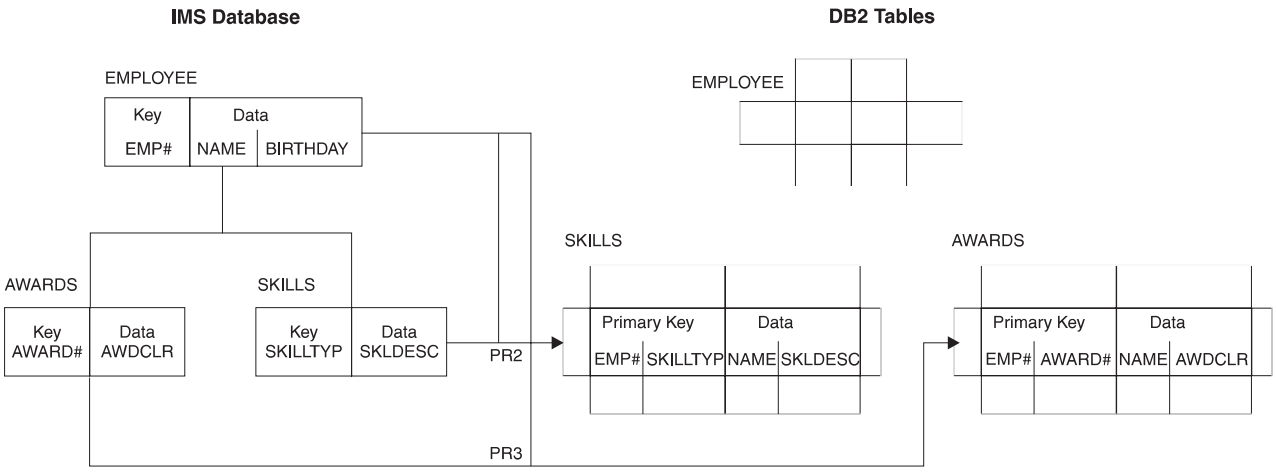


Figure 12. Denormalization of data with PATH Data

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 66, 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. And propagation can fail.
   b. Uniquely identify each parent/ancestor segment contributing modifiable PATH data to the propagation request. Identify segments by combining fields[4]mapped by the propagation request and located in either:
      - The parent/ancestor
      - A segment higher in the hierarchy

      Figure 13 on page 51 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.
      - SEG1 occurrences be uniquely identified through a combination of DATA1 and KEY1 values.

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

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 13, 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

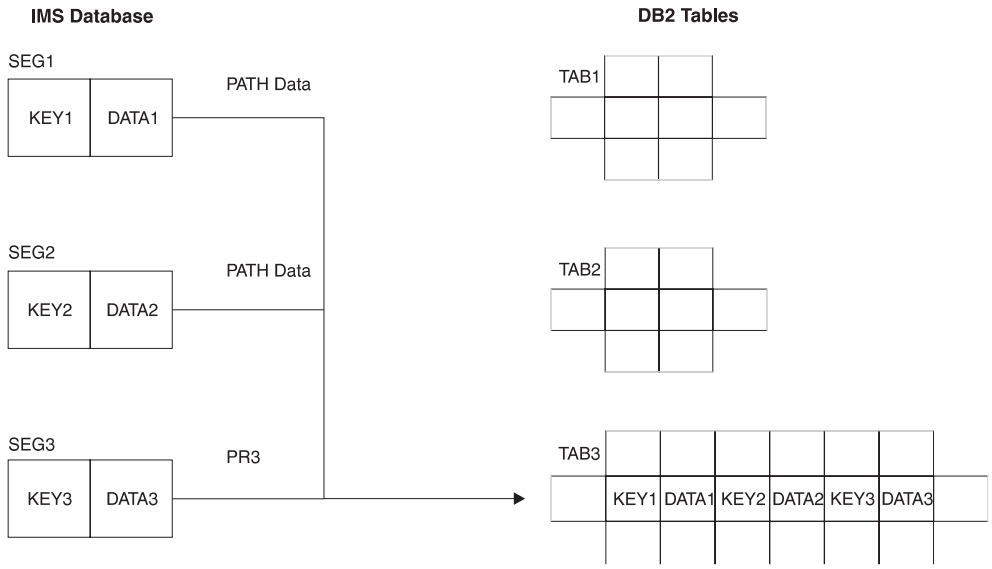


Figure 13. Identifying parent/ancestors contributing modifiable PATH data to PR3

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, they are not supposed to change their values.

In the example shown in Figure 14 on page 53, 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 70.

   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 14 on page 53, 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 14 on page 53, 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.

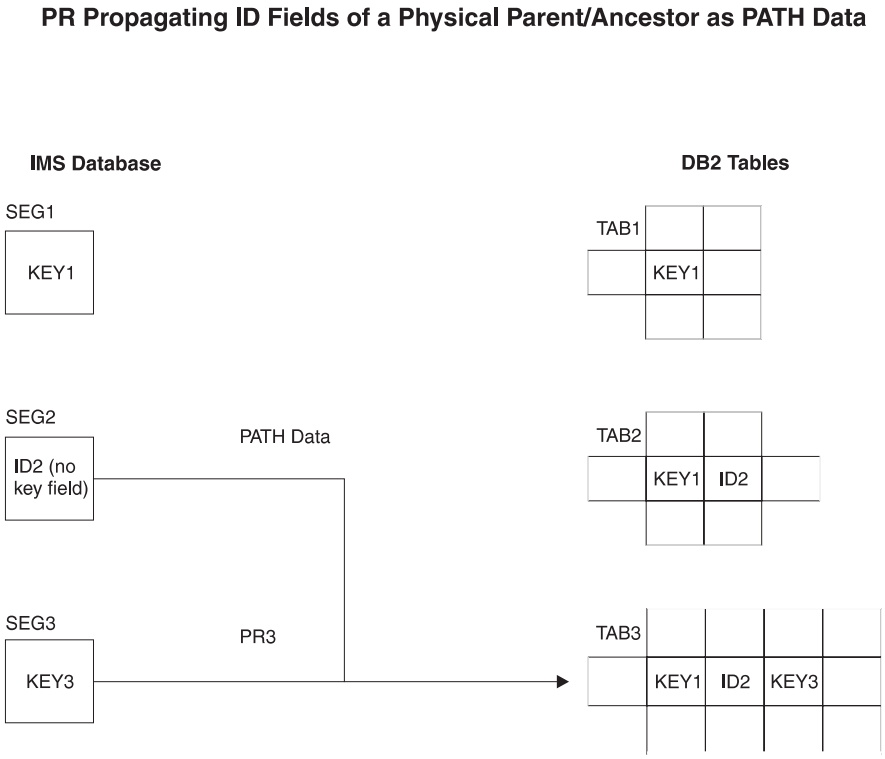


Figure 14. PR propagating ID fields of a physical parent/ancestor as PATH data

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. For a description of the limited use of the WHERE clause for DB2-to-IMS synchronous propagation, see “Selective propagation using the WHERE clause ” on page 54.

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 15 on page 54 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 15 on page 54, 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 15 would simply be reversed.

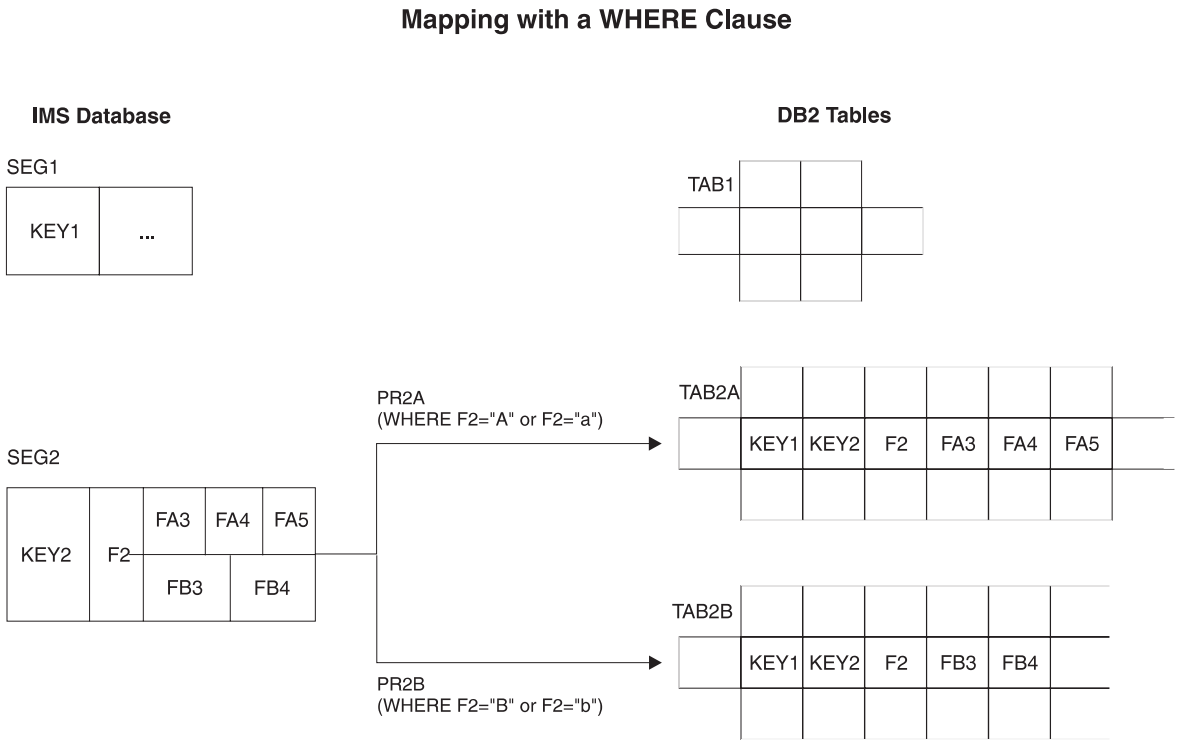


Figure 15. Mapping with a WHERE Clause

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, thereby providing selective DB2-to-IMS synchronous propagation. A WHERE clause specified during DB2-to-IMS synchronous propagation only calls for IMS DPROP to verify that the data mapped from DB2 to IMS satisfies the conditions specified in the WHERE clause. If it does not, IMS DPROP considers this an error unless DB2-to-IMS synchronous propagation is suppressed by an optional Segment exit routine.

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 you can include in the WHERE clause

In the WHERE clause, you can include:

- IMS *key fields* or sub-fields 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 aborts 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[5].
    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 that you cannot include 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.

5. 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[6]. 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 you can use 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*, SC27-1210 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.

6. 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 sub-fields 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 4. Propagation guidelines, rules, and restrictions

This chapter 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 completed and rules should be considered.

Topics included in this chapter 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 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 32.)

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.

	Which IMS delete rule is supported?			
Segment	**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:

- Bidirectional relationships, IMS DPROP has no preference between a PHYSICAL and LOGICAL delete rule (synchronous)
- 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 32). 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.

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.

This section describes some of the 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 16.

Defining Variable-Length Segments

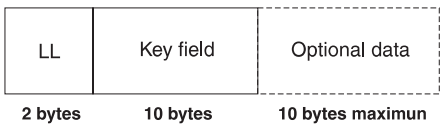


Figure 16. Defining variable-length segments

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

Propagating a subset of the columns in a table is straightforward for one-way DB2-to-IMS synchronous propagation. However, for IMS-to-DB2 propagation, during insert operations, propagation sets the values of the non-propagated columns to a default or a null value.

This section describes:

- How non-propagated columns are handled by IMS DPROP during IMS-to-DB2 propagation
- Requirements and recommendations for one-way IMS-to-DB2 propagation and two-way synchronous propagation

Assigning values to non-propagated 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 re-create 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 non-propagated 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.

Additional considerations for one-way IMS-to-DB2 propagation: Non-propagated 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 136 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 a given DB2 table. You can propagate with multiple propagation requests to the same table using a user mapping case but you cannot propagate to the same table with both user mapping and generalized mapping cases.

Propagating one segment to multiple tables

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

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

If you are implementing one-way IMS-to-DB2 propagation with PRTYPE=L, you can propagate one segment to multiple tables.

PRTYPE=E and DB2-to-IMS synchronous propagation

Generally, you cannot create multiple PRTYPE=E propagating the same segment to or 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 by only one PRTYPE=E. However, the containing segment can be propagated by another PRTYPE=E.
- IMS segments propagated by multiple PRTYPE=Es that specify a WHERE clause. One given segment occurrence should satisfy the WHERE clause of only one of the propagation requests. All the propagation requests must belong to the same mapping case.

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 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. [7]A number of IMS DPROP control statements support a PRSET= keyword. For example, with synchronous propagation, you can use SCU statements to activate, deactivate or suspend propagation requests. 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 using propagation request sets

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

7. 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.

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 only one table, whose qualified name is defined in the propagation request.

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 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 one of multiple, identically structured tables that have the specified unqualified table name.

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 143.

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

You should define the target DB2 tables without referential integrity constraints. By doing so, you:

- Allow a higher throughput
- Simplify the use multiple concurrent Apply program occurrences
- Avoid Apply failures caused by referential integrity enforcements

The definitions of referential integrity constraints do not make a lot of sense because the target DB2 tables are not supposed to be updated by user-provided logic or application programs. They should be updated only by IMS DPROP.

Defining unique indexes

To avoid propagation failures, observe IMS DPROP rules when defining unique DB2 indexes and doing IMS-to-DB2 propagation.

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.

The following sections discuss unique DB2 indexes and one-way IMS-to-DB2 propagation.

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.

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 19 on page 78 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.

Overview of the 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 previous IMS DPROP restrictions for PRTYPE=L have been eased in following releases. Even though some of the wording has changed, DPROPNR 1.2 and IMS DPROP 3.1 key mapping rules for PRTYPE=L are upwardly compatible with DPROPNR 1.1 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 2.1 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 1.1, you can still define PRTYPE=F in IMS DPROP 3.1.

Rules for PRTYPE=E (Extended Function)

PRTYPE=Es have strict key mapping rules but also provide more function than PRTYPE=L. Figure 17 on page 73 and Figure 18 on page 75 illustrate the key mapping rules for PRTYPE=E. Key mapping rules for PRTYPE=E are discussed in the following sections.

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

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. 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 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 18 on page 75, 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.
7. If you are doing DB2-to-IMS synchronous propagation of segments without a unique IMS key field, an IMS insert rule of HERE is treated by IMS DPROP as an insert rule of FIRST.

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 is not supported for MQ-ASYNC propagation.

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 187 and the*IMS DataPropagator for z/OS Reference*, SC27-1210.

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 17 on page 73 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.

Mapping Unique IMS Fully Concatenated Keys to DB2 Keys with PRTYPE=Es (Ideal Case)

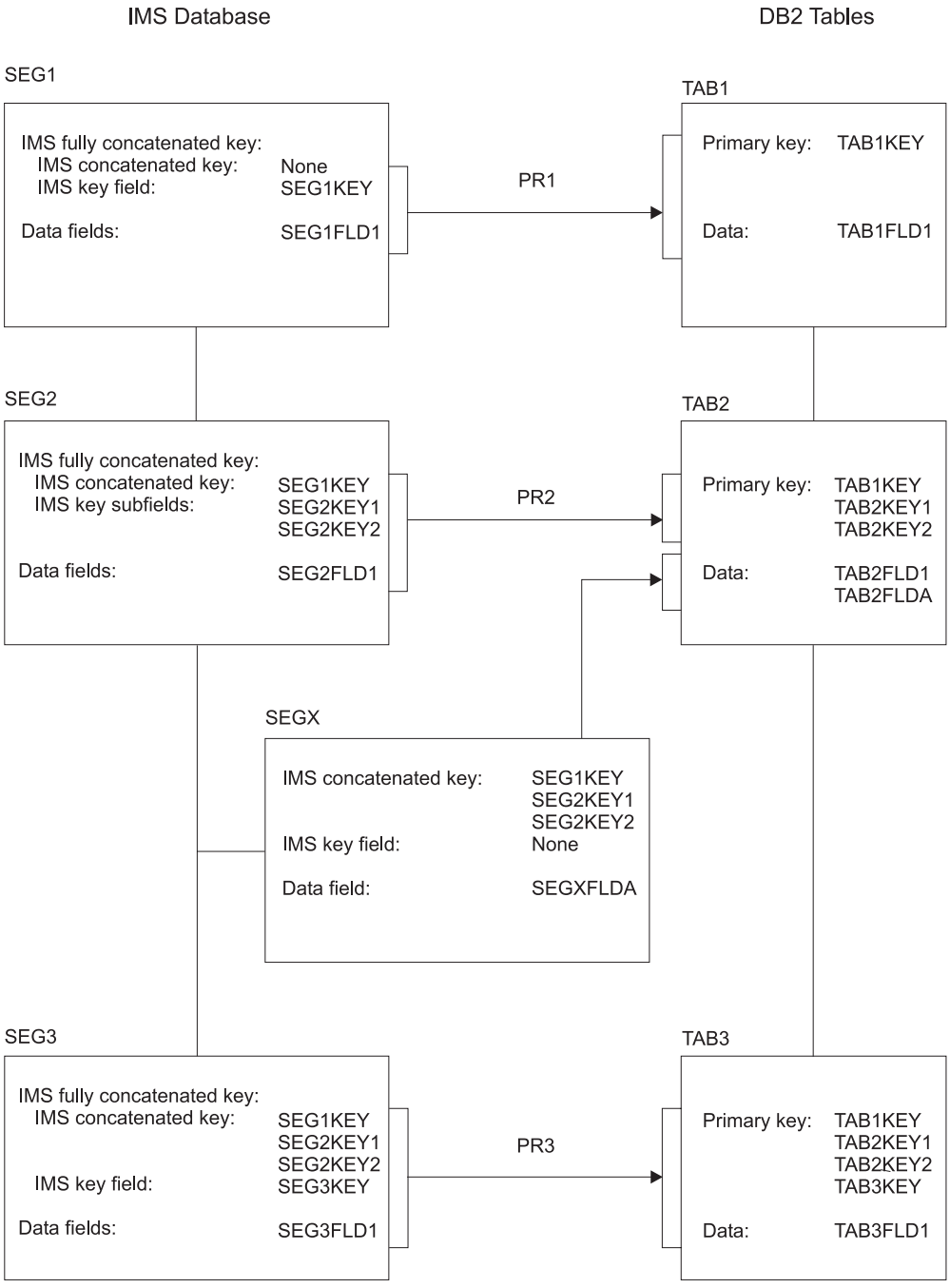


Figure 17. Mapping unique IMS fully concatenated keys to DB2 primary keys with PRTYPE=Es (ideal case)

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 sub-fields 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.
- 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 18 on page 75 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)

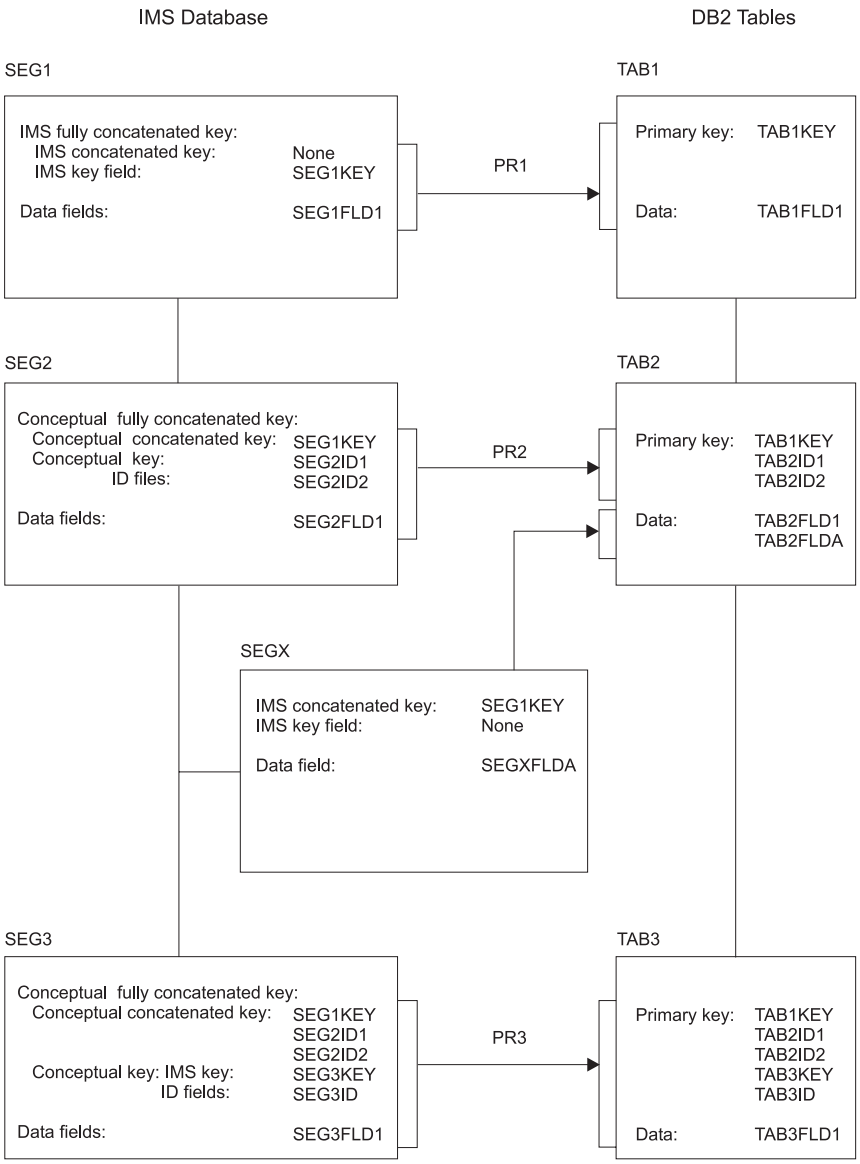


Figure 18. Mapping unique conceptual fully concatenated keys to primary DB2 keys with PRTYPE=E (non-ideal case)

Figure 18 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 TAB1KEY of TAB1 is the propagated unique IMS fully concatenated key of SEG1.
  - 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.

- 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 19 on page 78 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 IMS DPROP as a sequence of deletes and inserts of occurrences of the internal segment.

Key rule 4

Key Rule 4 is not supported for MQ-ASYNC propagation.

Key rule 5

Same as PRTYPE=E and L. See “Key rule 5” on page 72.

Key rule 6

Same as PRTYPE=E and L. See “Key rule 6” on page 72.

Example of mapping keys (PRTYPE=L)

Figure 19 on page 78 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.

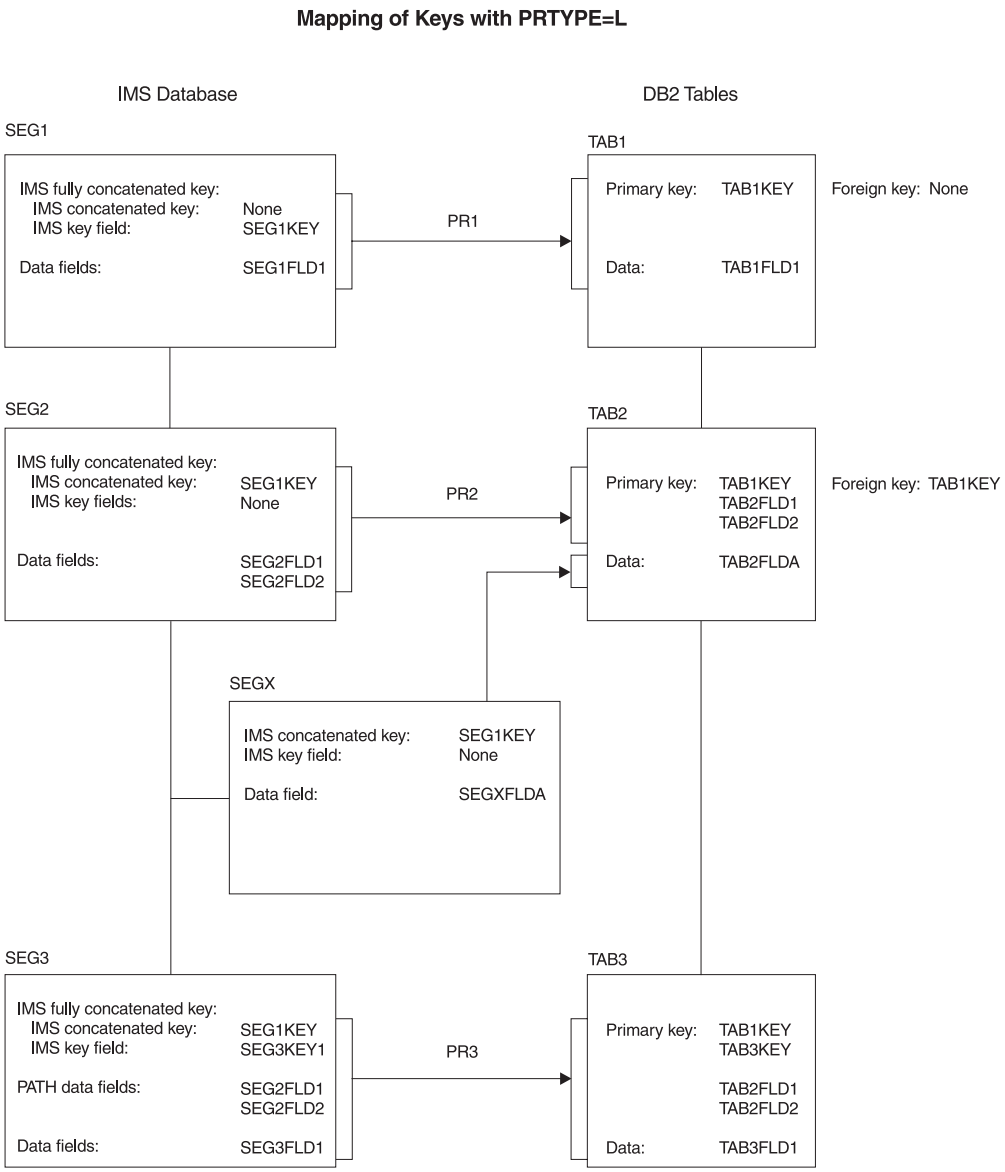


Figure 19. Mapping of keys with PRTYPE=L

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 19, 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.

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 4. 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 4. 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	
Not supported for MQ-ASYNC propagation.	
Key rule 5	
See description of key rule 5 on “Key rule 5” on page 72.	
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 5 on page 82. You can implement additional field formats and conversions using Field exit routines. For more information on Field exit routines, see the *IMS DataPropagator for z/OS Customization Guide*, SC27-1214.

You do not need to propagate every data field to 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 63.

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 nonnumeric data

Describing fields

To map IMS fields to 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 5 on page 82.

Table 5. 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.

During DB2-to-IMS synchronous propagation, IMS DPROP converts a DB2 null value to the default value for the data type of the IMS field, either zero, blank, or the current DATE, TIME, and TIMESTAMP. Or, you can write a Field exit routine that maps a DB2 null value to the value of your choice.

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). Refer to DB2 SQL Reference for a description of these formats.

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). Refer to DB2 SQL Reference for a description of these formats.

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 86 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 191.

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 84, 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.

During IMS-to-DB2 propagation mapping, 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 48.

Related Reading: Refer to the *DB2 for z/OS Administration Guide*, SC18-9840 for more information on normalization.

Chapter 5. Control information and environment

In addition to mapping your data, as described in Chapter 3, “Decisions affecting mapping and propagation ,” on page 31 and Chapter 4, “Propagation guidelines, rules, and restrictions ,” on page 59, you must prepare your operating environment.

This chapter describes:
- IMS DPROP control information
- 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 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 20 on page 90 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 RUP control block table contains one RUP Propagation Request control block (PRCB) for each propagated segment type.

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

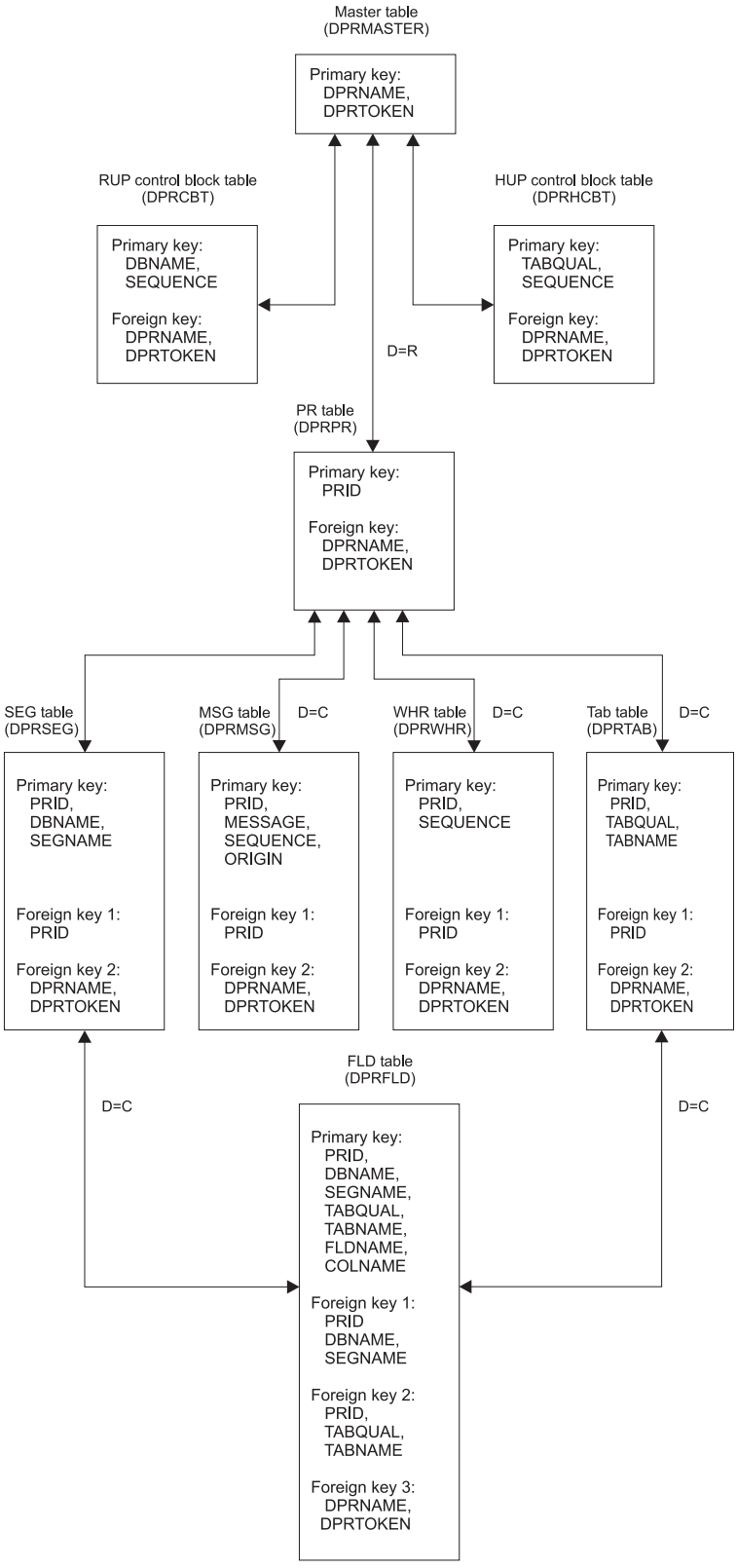


Figure 20. IMS DPROP Directory

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

The directory tables must be updated using 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.

IMS DPROP's use of 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 and IMS DPROP utilities.

VLF requirements

IMS DPROP creates and uses:

- One VLF object for each Capture Status file
- One VLF object for each Capture Transmission Specification file
- 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.

For information about how to provide SYS1.PARMLIB specifications to enable IMS DPROP to use VLF, refer to *IMS DataPropagator for z/OS Reference*, SC27-1210. For information on VLF itself, 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 DataPropagator for z/OS Apply[8] program using the SCU INIT VLF control statement. Use this statement after each IPL to initially populate VLF with the appropriate Apply[8] objects. The SCU INIT VLF control statement reduces the probability of DB2 enqueue conflicts for the IMS DPROP directory tables during propagation.

You can initialize and refresh the contents of the VLF class used by your IMS DataPropagator for z/OS Capture Exit routine using the CUT INIT VLF control statement. Use this statement after each IPL to initially populate VLF with the appropriate Capture objects.

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.

8. Only applicable with IMS-to-DB2 propagation.

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 *IMS DataPropagator for z/OS Reference*, SC27-1210.

IMS DPROP operational considerations

For MQ-ASYNC propagation, you must define at least two individual IMS DataPropagator for z/OS systems, one for Capture and another for Apply[9].

A Capture system can propagate IMS database updates to multiple DB2 target systems. However, each Apply[9] system is associated with one and only one DB2 system. If you want to propagate to multiple DB2 systems, one or more Apply[9] systems must be defined for each target DB2 system.

If you choose to perform MQ-ASYNC propagation with the Capture and Apply[9] systems on the same MVS image, IBM recommends that these IMS DataPropagator for z/OS systems be defined within the same IMS DataPropagator for z/OS environment.

Multiple IMS DPROP systems and environments

Each Apply[10] system has its own IMS DPROP directory. The Each IMS DPROP system is unaware of other IMS DPROP systems and other propagation requests defined in the directories of other IMS DPROP systems.

Each Apply[10] 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 propagation target tables
- Set of DB2 propagation target tables

Each Capture system also has its own set of VLF objects and classes.

9. Only applicable with IMS-to-DB2 propagation.

10. Only applicable with IMS-to-DB2 propagation.

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 IMS DPROP environment or belong to distinct IMS DPROP 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.

Typically, 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.

MQ-ASYNC propagation scenarios

This section illustrates scenarios for using one or more IMS DataPropagator for z/OS MQ-ASYNC propagation systems. In all the scenarios, the Capture and Apply systems can be on the same MVS image, different MVS images with shared DASD, or on remote MVS images.

Scenario 1

The scenario in Figure 21 shows an example of one of the least complex MQ-ASYNC scenarios with IMS-to-IMS propagation. In this scenario, there is a single capture system transmitting IMS database changes by means of MQSeries messages to a single IMS Apply on the same MVS image.

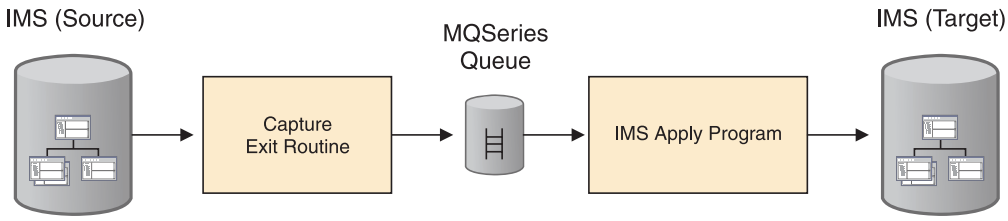


Figure 21. Propagation phase: MQ-ASYNC asynchronous IMS-to-IMS propagation (same MVS image)

Scenario 2

The scenario in Figure 22 shows another example of a less complex MQ-ASYNC scenario with IMS-to-DB2 propagation.

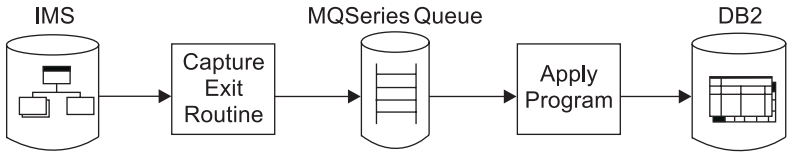


Figure 22. Propagation phase: MQ-ASYNC asynchronous IMS-to-DB2 propagation (same MVS image)

Scenario 3

The scenario in Figure 23 shows an example of the same scenario as shown in Scenario 1 for IMS-to-IMS propagation, except that in this scenario the Capture and IMS Apply are on different MVS images.

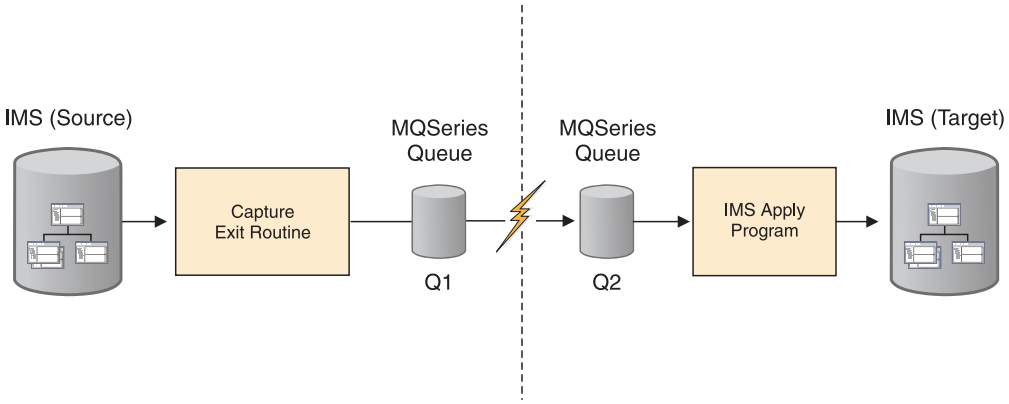


Figure 23. Propagation phase: MQ-ASYNC asynchronous IMS-to-IMS propagation (different MVS images)

Scenario 4

The scenario in Figure 24 shows an example of the same scenario as shown in Scenario 2 for IMS-to-DB2 propagation, except that in this scenario the Capture and IMS Apply are on different MVS images.

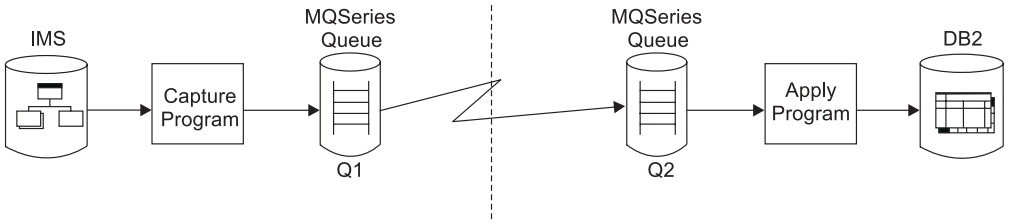


Figure 24. Propagation phase: MQ-ASYNC asynchronous IMS-to-DB2 propagation (different MVS images)

Scenario 5

The scenario in Figure 25 on page 95 shows a single IMS environment used as source for both IMS and DB2 propagation targets. One Capture system transmits (through MQSeries queues) to a discrete Apply system for DB2 target propagation and also to a discrete IMS Apply for IMS-to-IMS propagation. Both IMS Apply and Apply for DB2 are updating their IMS and DB2 targets on the same MVS image on which Capture is being initiated.

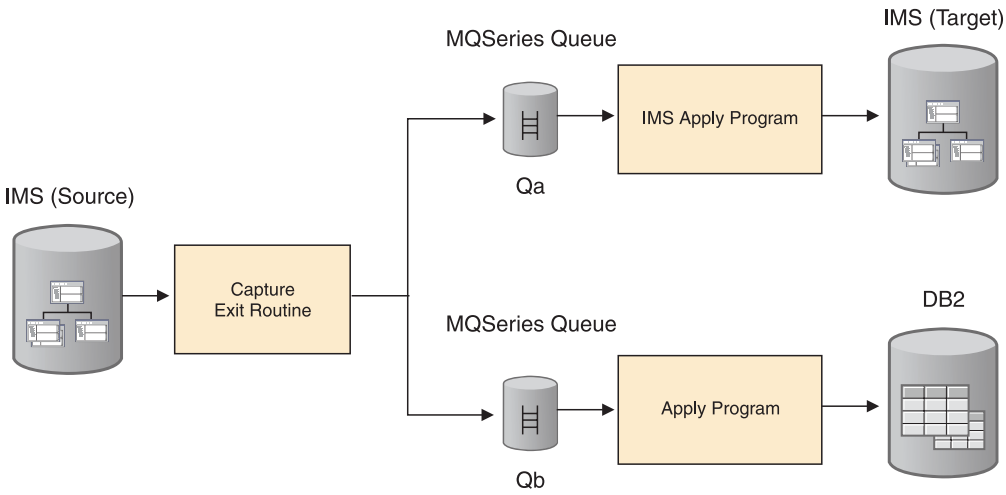


Figure 25. Propagation phase: one Capture system, one IMS Apply, and one Apply for DB2 system (same MVS image)

Scenario 6

The scenario in Figure 26 shows two sets of Capture and Apply systems, one set for a test system and another set for a production system. All the systems are shown on the same MVS image. The MQSeries queues are written to by the test and production Capture systems and are read from by the respective test and production Apply systems.

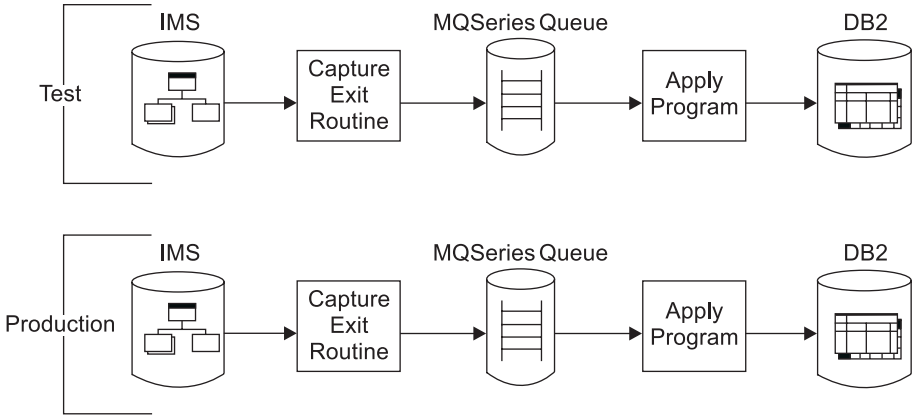


Figure 26. Propagation phase: two sets of Capture and Apply systems (same MVS image for Capture and Apply)

Scenario 7

The scenario in Figure 27 on page 96 shows the systems described in scenario 6, but here the Apply and Capture systems are on different MVS systems, although both individual Capture systems are on the same MVS image and both individual Apply systems are on the same MVS image.

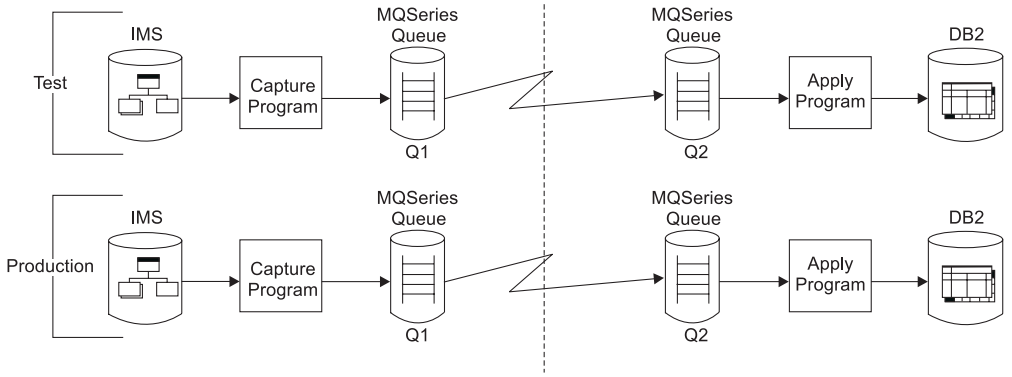


Figure 27. Propagation phase: two sets of Capture and Apply systems (different MVS images for Capture and Apply)

Scenario 8

The scenario in Figure 28 shows a single IMS environment that supports both test and production subsystems using a discrete Capture test system and a discrete Capture production system. In each instance, the test Capture and production Capture systems transmit (through MQSeries queues) to discrete test Apply and production Apply systems. Notice that all of these systems are running on the same MVS image and are updating their respective test and production DB2 systems.

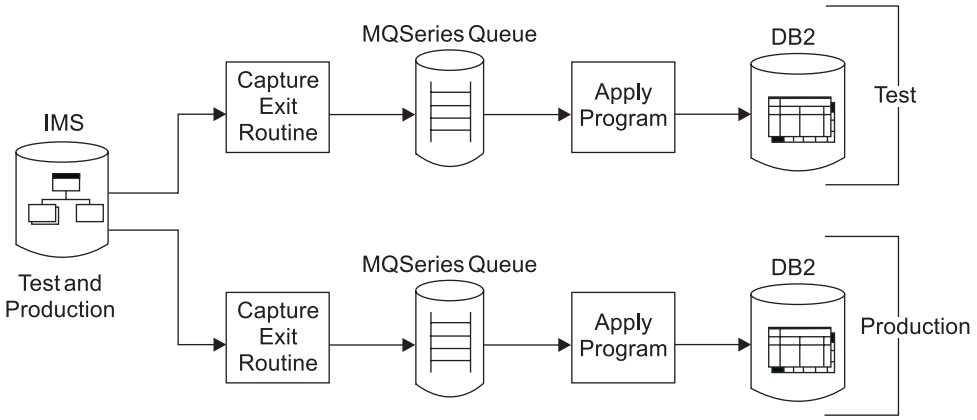


Figure 28. Propagation phase: discrete Capture and Apply systems for test and production environments (single MVS image)

Scenario 9

The scenario in Figure 29 on page 97 shows a single and separate test IMS environment sending updates to a discrete test Capture system and a separate production IMS environment sending updates to a discrete production Capture system. In each instance, the test Capture and the production Capture systems transmit messages (through MQSeries queues) to discrete test and production Apply systems on the same MVS image. In this scenario, both test and production Apply systems target the same DB2 system. All the systems in this scenario are running on the same MVS image.

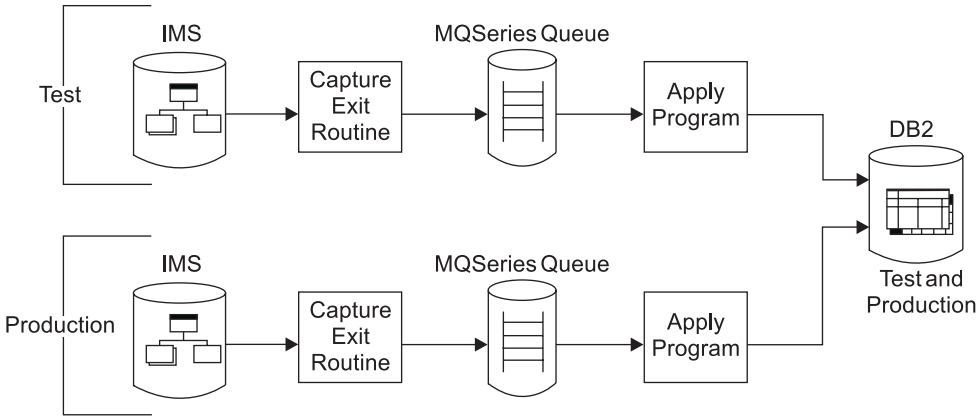


Figure 29. Propagation phase: two IMS systems, Capture systems, and Apply systems with single DB2 system (single MVS image).

IMS environment

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

- Use of DBRC
- IMS inserts in load mode
- Database updates with IMS utilities

Use of DBRC

When performing MQ-ASYNC propagation, IBM recommends that the source IMS databases be registered to Database Recovery Control (DBRC). Doing so facilitates coordinated establishment of IMS databases in read-only status using IMS DataPropagator for z/OS utilities.

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

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

IMS inserts in load mode

IMS DataPropagator for z/OS in MQ-ASYNC mode will propagate inserts performed with PCBs having PROCOPT=L or LS specified if the PROP LOAD option is specified through the //EKYIN DD statement. See the *Reference*, SC27-1210 for more details.

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.

CICS environment

MQ-ASYNC Capture does not support the capture of updates to IMS databases made through CICS transactions. BMPs and IMS batch updates, however, are supported.

Part 3. Setting Up for Data Propagation

These topics cover the setup phase of data propagation, including loading and extracting data:

- Chapter 6, “MQ-ASYNC initial setup tasks,” on page 103 describes the initial setup tasks to support MQ-ASYNC propagation.
- Chapter 7, “Setting up systems for MQ-ASYNC propagation,” on page 107, describes the steps involved in setting up IMS and DB2 for IMS DPROP and MQSeries asynchronous propagation, such as creating propagation groups and establishing start conditions.
- Chapter 8, “Defining and changing propagation requests ,” on page 115, describes how to define, change, and delete propagation requests.
- Chapter 9, “Granting privileges and authorizations for DB2 objects ,” on page 131, discusses granting authority for DB2 objects such as the IMS DPROP directory, MVG input tables, and DB2 plans.
- Chapter 10, “Binding and administering plans ,” on page 139, describes binding DB2 plans and administering DB2 plans.
- Chapter 11, “LOG-ASYNC and MQ-ASYNC coexistence and conversion for IMS-to-DB2 propagation,” on page 147 explains scenarios where LOG-ASYNC and MQ-ASYNC can coexist and also how to convert from LOG-ASYNC propagation to MQ-ASYNC propagation.
- Chapter 12, “Extracting and loading data ,” on page 149, explains how to extract data from an IMS database and load it into DB2 tables.

Chapter 6. MQ-ASYNC initial setup tasks

The following subjects are discussed later in this chapter as some of the tasks to be consider during the initial setup of the IMS DataPropagator for z/OS MQ-ASYNC propagation:

- Set up application programs involved in MQ-ASYNC Capture.
- Set up JCL of job step involved in MQ-ASYNC Capture.
- Set up the number of Apply program occurrences, the number of MQSeries queues, the definition of PRSTREAMs in EKYTRANS file.
- Set up MQSeries-related activities.

Application programs involved in MQ-ASYNC Capture

Application programs that plan to participate in MQ-ASYNC propagation changes should be aware of the following:

- MQSeries Queue Managers must be active and MQSeries queues used by EKYMQCAP must have been defined and must be ready to use.
- IMS batch job steps updating daatabases propagated by MQ-ASYNC must be run with RRS=Y on the DLIBATCH and DBBBATCH JCL procedure. RRS (Resource Recovery Services) will coordinate the commit processing of IMS batch and MQSeries.
- IMS batch applications that update propagated IMS databases and that issue their own MQSeries calls, must link-edit their code with the CSQBRSTB batch stub provided by MQSeries.

  CSQBRSTB is the RRS batch stub that provides a RRS-based two-phase commit coordination with the IMS database updates. If the code has been previously link-edited with another one of the MQSeries batch stub, then:
  - The code must be link-edited again, this time with the CSQNRSTB.
  - Any MQCMIT (Commit) and MQBACK (Backout) calls must be removed from the application because with CSQBRSTB, Commits and Backouts are coordinated by RRS. Instead, issue IMS Checkpoint and Rollback calls.
- For IMS batch programs, either convert them to BMPs or use RRS.
- RRS must be ready to be used before IMS batch job steps attempt to propagate IMS database changes.
- For those IMS batch and BMP programs where performance is critical, an appropriate commit frequency (checkpoint frequency) is especially important. The use of the MQSeries DEFINE MAXMSGS command that defines the maximum number of messages that a program can write within a unit of recover also needs to be considered.
- MQ-ASYNC does not support IMS database updates performed in a CICS environment.
- MQ-ASYNC does not support IMS database updates performed in a ODBA environment
- MQ-ASYNC does not support IMS application programs issuing SETS/SETU calls or ROLS calls with tokens.

  For IMS batch programs that issue any kind of ROLS calls, consider converting them into BMPs or replace the ROLS calls by an abend.
- When it is not required, avoid specifying the PATH option on the EXIT= keyword.
- Both synchronous and MQ-ASYNC must be part of the same IMS DPROP environment if updates to IMS databases by both are allowed in an IMS batch region or IMS online dependent region.

JCL of job steps involved in MQ-ASYNC Capture

The Capture component of MQ-ASYNC, implemented in IMS Data Capture exit routine EKYMQCAP, accesses various files in IMS batch regions or in IMS dependent online regions where it gets control as the result of the IMS database updates of the IMS application programs. The Capture component runs in the same environment as the updating IMS application programs. The JCL for the Capture job steps should be adapted to reflect the requirements of IMS DPROP, of MQSeries, and, for batch job steps, of RRS. They are:

```
//EKYMQST DD (Status file)
//EKYTRANS DD (Transmission Specification file)
//EKYIN DD (Optional, a secondary source of control information)
//JOBLIB DD
//STEPLIB DD
//EKYRESLB DD
//EKYPRINT DD
//EKYWTO DD
//EKYSNAP DD
//EKYTRACE DD
//EKYLOG DD
```

Related Reading: For a description of those files, see the*IMS DataPropagator for z/OS Reference*, SC27-1210.

Apply programs, MQSeries queues, and PRSTREAM definitions

In the simplest case, an installation might use only one occurrence of the Apply program. Often, however, installations will have multiple occurrences of the Apply program.

Installations having more than one Apply program occurrence might do so for the following reasons:

- When propagating data to DB2/390 on different systems or Sysplexes, at least one Apply on each System/Sysplex is needed. If high throughput is required, one set of Apply program occurrences is then needed.
- When propagating data both in near-real time mode and in point-in-time mode, at least one or one set of Apply program occurrences for each mode is needed.
- When propagating data to reflect different points in time, at least one or one set of Apply program occurrences for each different point-in-time propagation is needed.
- When propagating data where operational flexibility for scheduling the Apply process is desired (for example, applying the database changes to the set of DB2 target tables belonging to Application A while the set of target tables belonging to Application B are unavailable), at least one or one set of Apply program occurrences for each application is needed.

Because one MQSeries Queue cannot be read concurrently by more than one Apply program occurrence and that one Apply program occurrence can read only from one MQSeries Queue, the number of Apply program occurrences will affect the number of MQSeries queues that will need to be defined and reserved for the exclusive use of MQ-ASYNC.

Because each propagation data stream flowing from the Capture component to the Apply component is sent in its entirety to one MQSeries queue of the target system, the number of Apply program occurrences and MQSeries queues affect the

PRSTREAM definitions in the //EKYTRANS File of the Capture component. Conversely, the design for the PRSTREAM definition in the //EKYTRANS File will influence the planning of the number of Apply program occurrences and the number of MQSeries queues.

For the syntax and the use of the parameters for the PRSTREAM control statement, see the *IMS DataPropator for z/OS Reference*, SC27-1210.

Be aware that multiple PRSTREAMs might flow to the same MQSeries queue and might be processed by the same Apply program. The granularities of PRSTREAM at the DBD level, at the SEGMENT level, and at the KEYRANGE level are important.

Review the examples of PRSTREAM definitions shown "The Transmission Specification file" on page 9. They cover:

- Example for just one PRSTREAM
- Example for two PRSTREAMs
  - The first PRSTREAM is for two databases at the database level
  - The second PRSTREAM is for two databases, one at the database level and another at the SEGMENT level.
  - Example for one PRSTREAM with specification of KEYRANGEs.

It might be necessary in some situations to change PRSTREAM definitions by splitting one PRSTREAM into multiple PRSTREAMs or merging multiple PRSTREAMs into one PRSTREAM. If this becomes necessary, make sure that:

- There are no source update activities. Database updates operations should be quiesced, for example, through the use of the READON SCU control statement.
- In the case of moving database changes of specific databases or Segments from one PRSTREAM into another PRSTREAM, the Apply program must have completed the processing of the database changes of these databases/SEGs that have been put into the MQSeries queues with the old PRSTREAM definitions before starting the process of the database changes of these databases/SEGs that are being put with the new PRSTREAM definitions.

The **PR**= and **PRSET**= keywords of the EXCLUDE/INCLUDE parameters of the Apply control statement can be used to limit the Apply processing to specific PRs or PRSETs.[11]

MQSeries-related setup activities

For an IMS DataPropagator for z/OS MQ-ASYNC system, you must decide whether to share an existing MQ system or to create a new MQ system. The Queue Manager should be defined without Dead Letter queues.

MQSeries is extremely robust and reliable and has a very high availability. Proper MQ-related administration and operation procedures should be in place in IMS DataPropagator for z/OS MQ-ASYNC users' installations.

The following are recommendations:

- The region error option (REO) of the SSM entry in IMS.PROCLIB should be specified as an **R**.

11. INCLUDE/EXCLUDE is supported only with IMS-to-DB2 propagation.

REO specifies the region error option to be used if an IMS application tries to reference a non-operational external subsystem or if resources are unavailable at create thread time. The specification of R means that a return code will be passed to the application, indicating that the request for MQSeries services has failed.

- Define MQSeries queues to be used in the IMS DataPropagator for z/OS MQ-ASYNC operations as exclusively reserved for the use of IMS DataPropagator for z/OS.
- Define those queues with the Persistent attribute.
- Estimate the storage required for the MQSeries queues with proper considerations given to the peak activities as well as the possibility that some resources might not available for long period of times. Examples of this might be a target system or target table not available for a whole weekend for maintenance reasons. The MQSeries queues should be sufficiently large to absorb all messages that are being created during this time. Monitor MQSeries queue size and usage with MQSeries specific tools.
- Estimate and make available the required TP bandwidth.

Chapter 7. Setting up systems for MQ-ASYNC propagation

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

- Creating or changing DBDs
- Creating DB2 tables, if needed
- Binding plans
- 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*, SC26-8035. 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:

- 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 ” on page 108 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 59. You might need to modify your delete rules, and make changes to the logic of your application programs.
- You can specify a VERSION= keyword in the DBD macro. Depending on IMS DPROP generation options specified during installation, either the RUP might validate the version to reduce errors resulting from inconsistent DBD libraries and IMS DPROP directories.

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.

After a DBD is changed, you must run ACBGEN if the database is referred to in an online IMS environment. Refer to *IMS Utilities Reference: System*, SC18-7833 for more information about the ACBGEN and DBDGEN processes.

The following sections describe how you use the:

- EXIT keyword
- VERSION keyword

EXIT keyword

For IMS DPROP propagation, the DBD defining the physical database must include an EXIT keyword.

- IMS calls one or more IMS Data Capture exit routines with the captured data.
- Specify EKYMQCAP as the name of the IMS Data Capture exit routine for the MQ-ASYNC propagation.

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 109.

You can specify multiple exit names, so you can propagate the same segment with MQ-ASYNC (using EKYMQCAP). 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 EKYMQCAP as an exit name, so you can propagate the same segment asynchronously with both Log-ASYNC and MQ-ASYNC.

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 sub-options. 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

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 109. 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 109.

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 the *IMS/ESA Utilities Reference: System*, SC26-8035 and the IMS DPROP *IMS DataPropagator for z/OS Reference*, SC27-1210.

In this section, examples:

1 to 3 Show how to specify MQSeries asynchronous propagation of the same segment type with other forms of propagation.

The following examples apply to IMS DPROP or user asynchronous propagation.

Example 1 (only MQSeries asynchronous propagation)

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

Example 2 (any combination of MQ-ASYNC, Log-ASYNC, and user asynchronous propagation using the ACDC)

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

Shows how to propagate the segment type:

- Asynchronously with MQSeries Capture
- IMS log-based asynchronously with no cascade deletes

Example 3 (any combination of MQ-ASYNC, synchronous and user-async propagation)

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

Shows three different IMS Data Capture exit routines:

- EKYMQCAP propagates the segment asynchronously using MQSeries
- 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).

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 a 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.
- 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 DataRefresher.

Related Reading: For more information on the VERSION keyword of the DBD macro, refer to *IMS Utilities Reference: System*, SC26-8035 and the *IMS DataPropagator for z/OS Reference*, SC27-1210.

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 the *DB2 for z/OS Administration Guide*, SC18-9840 and *DB2 for z/OS SQL Reference*, SC18-9854.

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 requests.

Establishing the start conditions for IMS DPROP Asynchronous MQSeries propagation

Before you can start IMS DPROP asynchronous propagation, you must:

1. Use the SCU READON control statement to quiesce all IMS databases that are involved in IMS DPROP asynchronous propagation. Or you must take DEDBs offline.
2. Carry out a full extract of the IMS databases and load the data into the DB2 tables, as described in Chapter 12, “Extracting and loading data ,” on page 149.

   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.
3. Use the SCU READOFF control statements to restart all IMS databases that are involved in IMS DPROP asynchronous propagation. Or, you can bring DEDBs back on line.

Related Reading: For details of the SCU control statements, refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210.

Chapter 8. Defining and changing propagation requests

Creating a propagation request involves specifying the IMS fields (or data elements), IMS segments, DB2 columns, and DB2 tables involved in propagation. 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. For a description of the extract and load with DataRefresher DEM, see Chapter 12, “Extracting and loading data ,” on page 149.

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.

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 116.

- 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 116.
- 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 117.

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 118. 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 30 on page 120 illustrates the process. For complete information on using DataRefresher commands to define propagation requests, refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210. 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 a 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 a 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 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*, SC27-1210 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*, SC27-1210, 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, 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 125.

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 5, “Control information and environment,” on page 89.

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.
- MVG updates the master timestamp field in the master table in the IMS DPROP directory to signal changes to the RUP. If the RUP detects changes to the directory, it refreshes 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, the 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 the 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 30 on page 120.

Refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210 for more detailed information and examples.

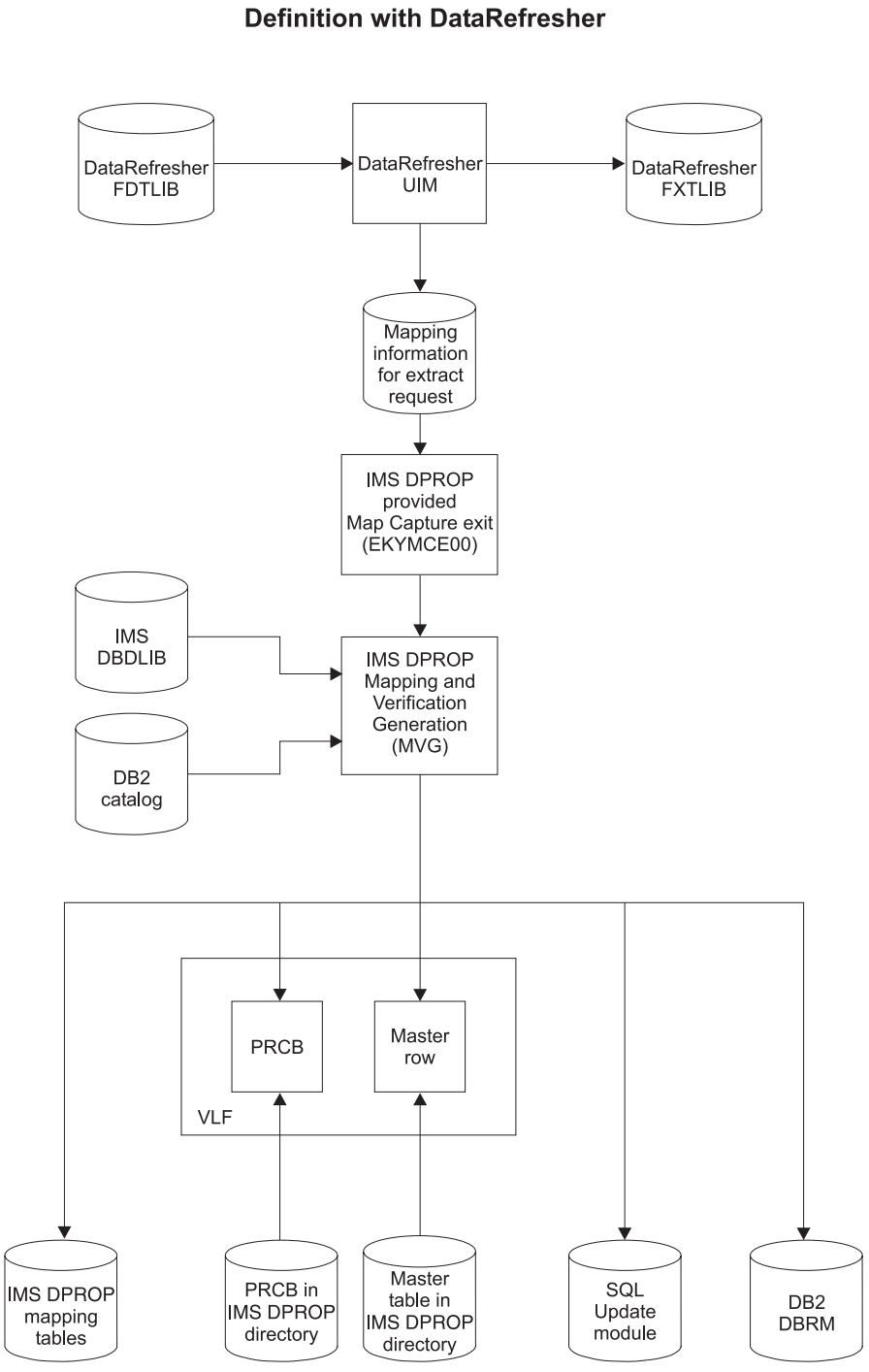


Figure 30. Propagation request definition with DataRefresher

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.

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*, SC27-1210.

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

For more details on the MVG input tables and their columns, refer to the *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 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 target DB2 table. Generalized mapping allows only a single propagated table for each propagation request. A target 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.

Executing the MVGU

Once you have provided the necessary propagation request information in the MVG input tables, you should execute 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 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 the RUP. If the 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, the 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 31 on page 124.

PR Definition with MVG Input Tables

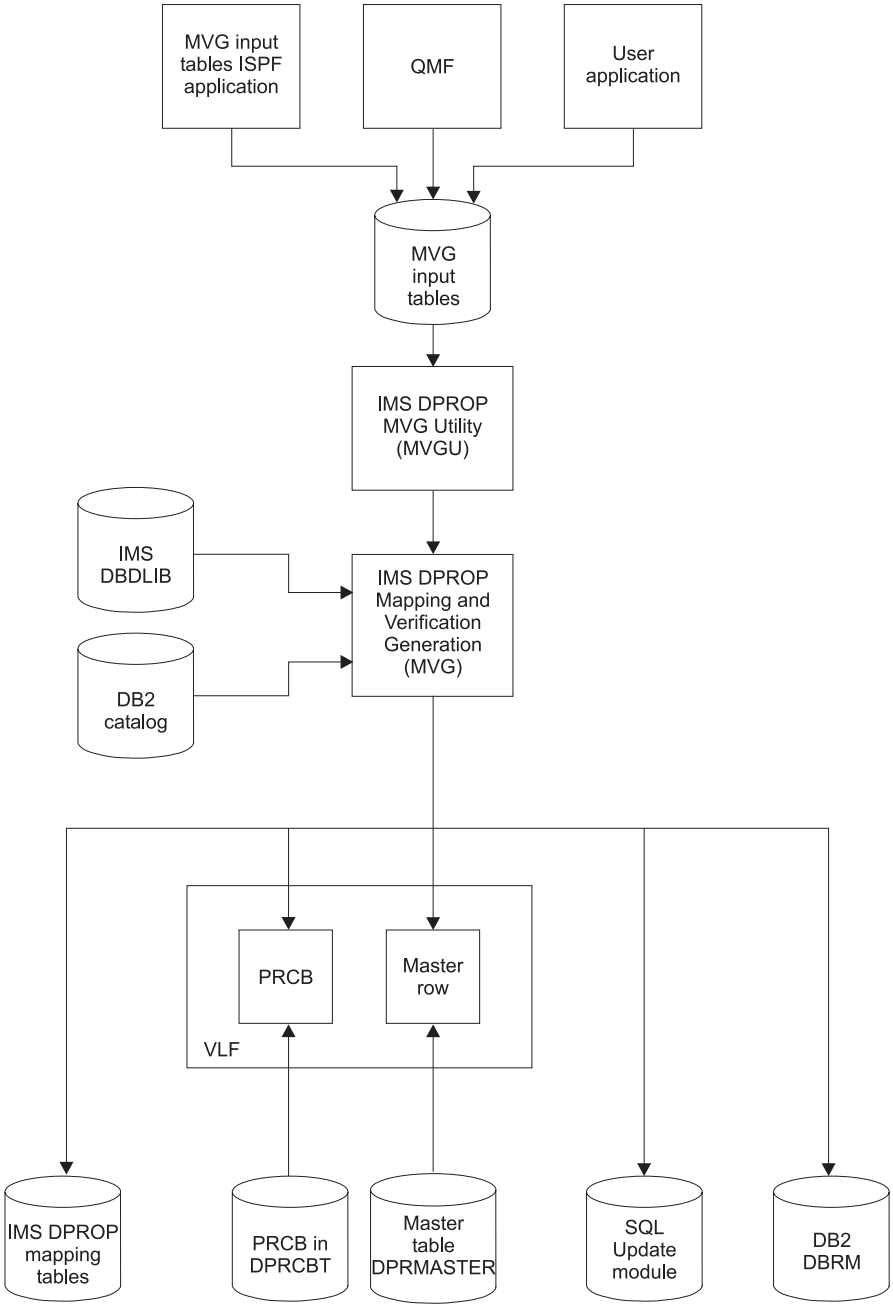


Figure 31. Propagation request definition with MVG input tables

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 DPROP Reference.

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 65.

ERROPT—Error option

Specifies the error option (BACKOUT or IGNORE) to be taken when a propagation request fails.

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*, SC27-1210.

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 doesn't 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.

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.

For detailed information on how to code the DELETE command, see the *IMS DataPropagator for z/OS Reference*, SC27-1210.

Replacing a propagation request

To change a propagation request in the IMS DPROP directory, the propagation request must be re-created 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.

- 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*, SC27-1210.

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.

Related Reading: For detailed information on how to code the RECREATE control statement, see the *IMS DataPropagator for z/OS Reference*, SC27-1210.

Revalidating propagation requests

The IMS DPROP 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 9. 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 perform 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 10, “Binding and administering plans ,” on page 139.

Granting privileges for IMS DPROP collections

As part of the IMS DPROP 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 133.

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

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 10, “Binding and administering plans ,” on page 139.

Running IMS DPROP utilities

Running a 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 execute 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 rebinding 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 execute the IMS DPROP utilities front end applications

The CCU 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 executes any of these front end applications needs the SELECT privilege for the IMS DPROP directory tables and for the MVG input tables.

Propagated tables 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

Granting table privileges for propagated tables

This section provides considerations for granting privileges for one-way IMS-to-DB2 propagation.

When doing one-way 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 re-synchronizing the DB2 copy after propagation has been suspended

You should grant the SELECT privilege for propagated tables to those who:
- Use decision support systems
- Query propagated tables
- Run the CCU

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 32 illustrates the concept of updating non-propagated columns.

Columns You Can Update in Propagated DB2 Tables

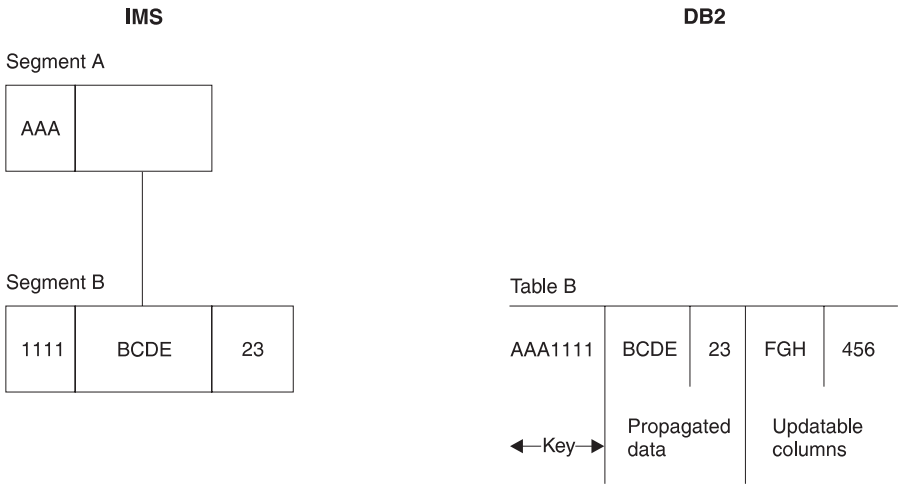


Figure 32. Columns that may be updated in propagated DB2 tables. The columns containing FGH and 456 can be updated through a view. The key and propagated data should be read-only.

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 need to provide your own program in order 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 perform 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 EXECUTE privilege for the entire collection ID to future owners of plans of propagating applications, you must grant EXECUTE privilege for individual packages. You might also need to grant BIND and COPY privilege for individual packages.

Chapter 10, "Binding and administering plans ," on page 139 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 EXECUTE privilege for the entire collection ID to future owners of plans of propagating applications, then you must grant EXECUTE privilege for individual packages.

Chapter 10, “Binding and administering plans ,” on page 139 has additional information about binding packages.

Chapter 10. Binding and administering plans

This chapter describes binding DB2 plans of the Apply program.

You must bind DB2 plans for whatever calls IMS DPROP because IMS DPROP makes SQL updates and the SQL reads the IMS DPROP directory, which is composed of DB2 tables. 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 the package bind facility

You can use the DB2 package bind facility for the DB2 plans of your propagating applications and Apply 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:

- 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 package 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 do 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 the Apply program and other things 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 33 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 33. 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 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 34 on page 142 is a sample job stream for binding a plan of an Apply program and a propagating application. 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 34. 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 Apply 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 IMS DPROP read-only collection. If necessary, ask your system administrator which IMS DPROP system and read-only collection your plan is supposed to work with.

   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 IMS DPROP 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 have already been 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 Apply program, you either provide different ENABLE specifications or omit the ENABLE specifications.

Binding plans without bind package

This section describes binding the plans of application programs and the Apply programs without using the DB2 package bind option.

Binding the Apply program

The DB2 plan for the Apply program consists of the DBRMs of IMS DPROP modules as well as the original DBRMs of the Apply program. If you are doing MQ-ASYNC propagation, the Apply 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 Apply program's access to DB2 tables

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 35 on page 144 shows the two-step BIND process when you create

aliases or synonyms before bind.

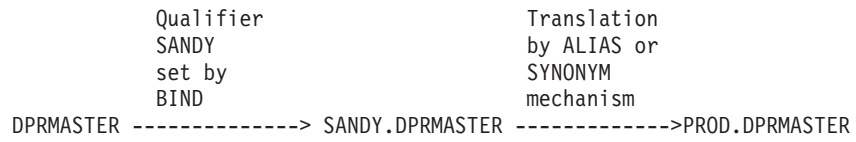


Figure 35. Two-Step BIND Process

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 36.

```
CREATE ALIAS SANDY.DPRMASTER FOR PROD.DPRMASTER
```

Figure 36. 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 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 37. 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 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.

Chapter 11. LOG-ASYNC and MQ-ASYNC coexistence and conversion for IMS-to-DB2 propagation

Coexistence

MQ-ASYNC can coexist with LOG-ASYNC. They can either have their own DPROP directories or they can use the same DPROP directory concurrently. Sharing a DPROP directory simplifies a gradual conversion from LOG-ASYNC to MQ-ASYNC.

When propagating to staging tables, LOG-ASYNC and MQ-ASYNC do not set the IBMSNAP_COMMITSEQ to exactly the same value. LOG-ASYNC uses the STCK value from the IMS Commit Log record while MQ-ASYNC uses the STCK value of the IMS database change. Therefore, if both LOG-ASYNC and MQ-ASYNC propagate IMS updates of the same IMS UOW to staging tables, it will appear that the updates propagated by LOG-ASYNC and MQ-ASYNC belong to different IMS UOWs.

IMS DPROP does not check whether the same PR is concurrently processed by both LOG-ASYNC and MQ-ASYNC.

- Often, this is not valid (for example, when both LOG-ASYNC and MQ-ASYNC propagate concurrently to the same target DB2 table).
- Some other times this is valid (for example if the PR has been defined with an unqualified name for the target DB2 table, and if LOG-ASYNC and MQ-ASYNC propagate to two different target DB2 tables having different table name qualifiers).

It is the responsibility of the user to avoid invalid concurrent processing of the same PR by LOG-ASYNC and MQ-ASYNC.

Conversion from LOG-ASYNC to MQ-ASYNC

If you are currently using LOG-ASYNC and want to propagate your IMS databases in near-real time, or for any other reasons, decide to convert to MQ-ASYNC systems, you can migrate either all your PRs or a subset of your PRs at one time. It is not necessary to convert all the PRs at the same time.

If you want to convert S your LOG-ASYNC systems to MQ-ASYNC systems, migrate your previous IMS DPROP release to IMS S DPROP V3.1 and exercise the migrated LOG-ASYNC systems under V3.1 before converting them to MQ-ASYNC systems.

The following are two possible conversion scenarios:

- Go through an Extract/Load process at the time of migration.
- Stop the LOG-ASYNC Receiver at a specific point in time when the affected IMS source DBs are quiesced, and then switch to the Apply program of MQ-ASYNC using an EXCLUDE UNTIL= control statement to start the apply process where the LOG-ASYNC Receiver stopped.

If you need to perform a fallback process, you can exploit the availability of:

- The STOPAT support of the MQ-ASYNC Apply program.
- The capabilities of the LOG-ASYNC to start the propagation at a specific point-in-time.

Chapter 12. Extracting and loading data

The first part of this chapter explains the process of extracting data from a source IMS database and loading it into a target IMS database. This process is part of IMS-to-IMS propagation. The second part of 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. This process is part of IMS-to-DB2 propagation.

For details on how to code an extract request for DataRefresher refer to the*IMS DataPropagator Reference*, SC27-1210. For details on how to code an extract request using your own program see *IMS DataPropagator Customization Guide*, SC27-1214.

Extracting and loading data for IMS-to-IMS propagation

This section explains the process of extracting data from a source IMS database and loading it to a target IMS database. This section presents:

- An overview of the extract and load process
- A description of doing the extract and load
- IMS DPROP asynchronous extract and load considerations

Overview of the IMS source extract and IMS target load process

After identifying the IMS databases to be propagated and identifying the data to extract from them, follow the steps below to extract the data and load it into target databases.

1. First, use the appropriate IMS commands to prevent updates to IMS source databases.
2. Next, extract data from IMS source databases.
3. Then, load data into IMS target databases.

For large databases, a substantial amount of time might be required to perform the following tasks:

- Perform an image copy of the IMS source databases.
- Extract data from IMS databases.
- Load the data into the IMS target databases.
- Perform an image copy of the loaded target databases.

You should ensure that the IMS source databases are not available during the extract phase. After data extraction is complete, updates may be resumed to IMS source databases with data capture activated using the IMS DPROP MQ-ASYNC Capture exit EKYMQCAP. When the IMS target databases have been loaded, the captured data may be applied to the IMS target databases using the IMS DPROP MQ-ASYNC IMS Apply program.

Performing extract and load for IMS-to-IMS propagation

Because no data transformation is required when implementing IMS-to-IMS data propagation, extracting and loading propagated data is relatively uncomplicated. The process can be performed using standard IMS database utilities of your choice. Two methods are outlined below.

Method 1

First, run the image copy utility when the IMS source database is offline from IMS. Then, using the IMS Recovery utility from the image copy of the source database to load the IMS target database.

Method 2

First, run the HD Unload while the source IMS database is offline from IMS. Then, use unloaded file with the HD Reload utility to load the IMS target database.

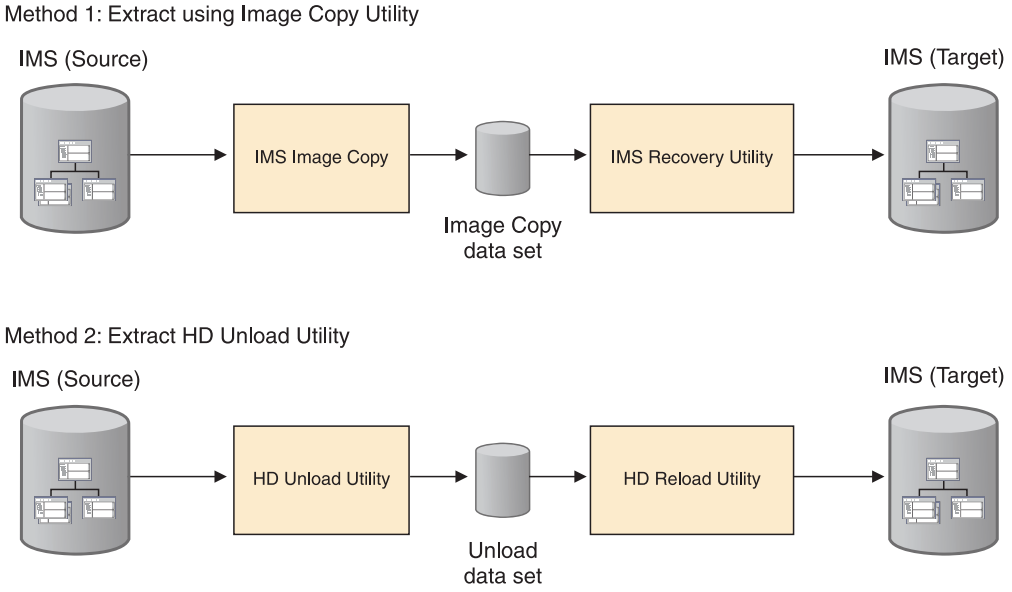


Figure 38. IMS-to-IMS extract using the Image Copy utility and HD Unload utility

Extract and load considerations for IMS-to-IMS propagation

Under normal circumstances, you probably maintain duplicate database copies of the IMS source and the IMS target. If this is the case, you must initially populate the IMS target copy prior to propagation, using your method of choice. The IMS source database must be quiesced in order to get an extract of the source with integrity. If updates are made during the extract process, data inconsistencies can occur.

Extracting and loading data for IMS-to-DB2 propagation

This section explains the process of extracting data from an IMS database and loading it into a target DB2 table using DataRefresher or your own program. This section presents:

- 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
- IMS DPROP asynchronous extract and load considerations

Overview of the IMS source extract and DB2 target load process

You usually perform extract and load after creating propagation requests. To extract data from IMS and load it into target DB2 tables:

1. First, prevent updates to IMS databases using the SCU or the appropriate IMS commands
2. 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.
3. 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.

At this point, you can activate the propagation requests with SCU and make the databases available for updates during synchronous propagation.

For large databases, a substantial amount of time might be required to perform the following tasks:

- Perform an image copy of the IMS databases.
- Extract data from an IMS database.
- Load the data into DB2 target tables.
- Build index entries (part of the DB2 load process).
- Perform an image copy of the loaded tables.
- Execute the RUNSTATS utility against the DB2 tables.

You should plan for the IMS databases being propagated not being available during the extract and load phase.

Preventing updates to IMS databases for IMS to DB2 propagation

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 some 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 following SCU statements:

1. 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 Utilities Reference: Database and Transaction Manager*, SC27-1308. For more information on the SCU, see the *IMS DataPropagator for z/OS Reference*, SC27-1210.

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.

Refer to IMS/ESA Operations Guide for specific information on the /DBD and /STA commands.

Performing extract and load with DataRefresher for IMS to DB2 propagation

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 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.

Refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210. for detailed information on how to code an extract request for DataRefresher.

The extract and load process is illustrated in Figure 39.

Extract and Load Process Using DataRefresher

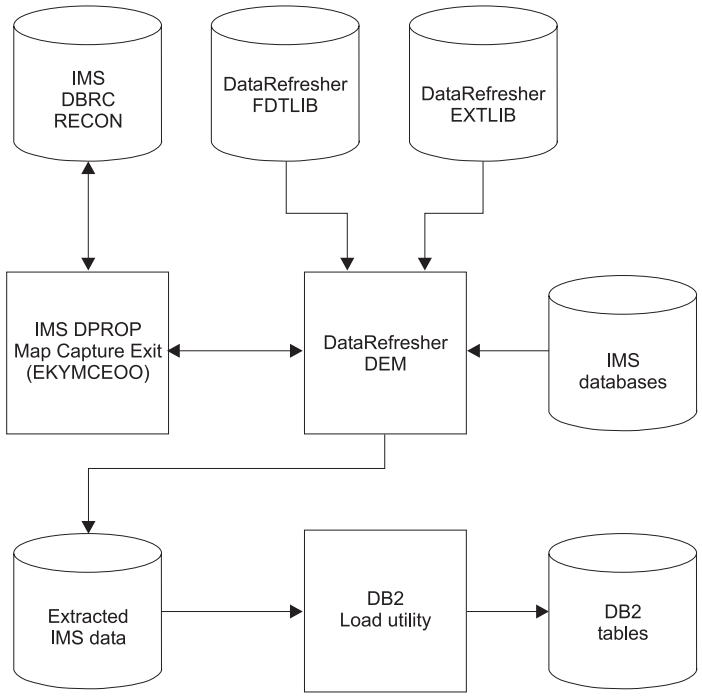


Figure 39. Extract and load process using DataRefresher. After the tables have been loaded, you should run the RUNSTATS utility and copy the tables.

Consider the following information when you use the DataRefresher DEM to extract data propagated by IMS DPROP:

- 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. For more information on this subject, refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210.
- 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. For a description of the restrictions, refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210.
- 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 for IMS to DB2 propagation

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 for z/OS Command Reference*, SC18-9844 and *DB2 for z/OS Utility Guide and Reference*, SC18-9855.

  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.

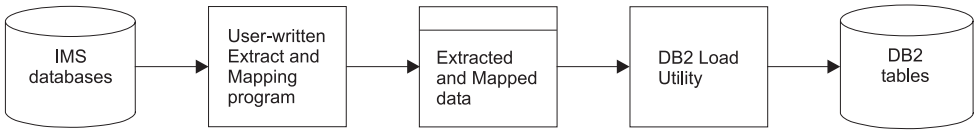
If you use methods 1 and 2, your programs must provide the IMS-to-DB2 mapping and conversion logic. The mapping and conversion must be compatible with IMS DPROP's mapping and conversion. If you want to use the CCU to verify data consistency, your mapping and conversion must be identical to that of IMS DPROP.

Figure 40 is an overview of the three methods of doing the extract and load using your programs.

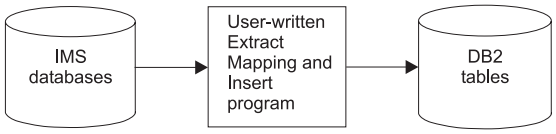
See the *IMS DataPropagator for z/OS Reference*, SC27-1210 for information on how to code an extract request using MVG input tables without DataRefresher.

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

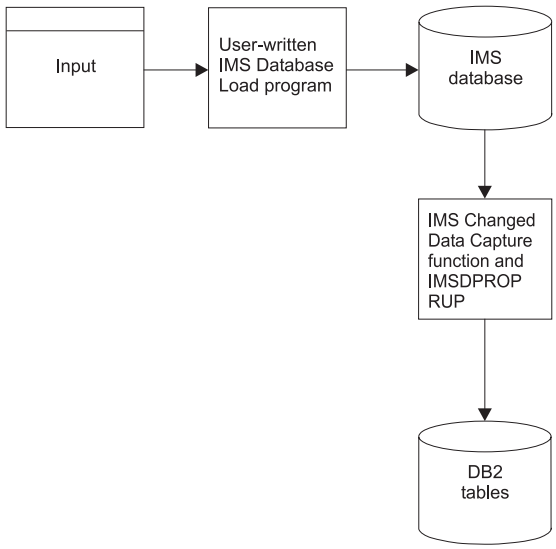


Figure 40. Extract and load process with user-written programs

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.

IMS DPROP Asynchronous MQSeries 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 READONLY, using the IMS DPROP SCU. Take DEDBs offline for the extract process.

Part 4. Propagating with MQ-ASYNC

These topics cover the propagation phase and maintenance and control phase of MQ-ASYNC:

- Chapter 13, “Performing MQ-ASYNC propagation ,” on page 161, describes normal processing and error control in an asynchronous environment.
- Chapter 14, “Propagating IMS data to staging tables ,” on page 173, describes propagation of IMS data to intermediate staging tables. The data in the staging tables can then be accessed by programs such as IMS DPROP for propagation to DB2.
- Chapter 15, “MQ-ASYNC considerations ,” on page 181, describes some of the many possible user scenarios and outlines some of the considerations that must be taken into account when you implement MQ-ASYNC.

- Chapter 16, “Verifying consistency between IMS and DB2 data copies (CCU),” on page 187, describes use of the Consistency Check utility (CCU) for verifying IMS and DB2 data consistency.
- Chapter 17, “Problem determination tools ,” on page 195, explains how to get information about various database objects and system activities using IMS DPROP’s auditing and tracing facilities, message table, and CCU.
- Chapter 18, “Performance and monitoring,” on page 201, discusses how to optimize performance in the IMS DPROP environment and how to monitor propagation using IMS and DB2 tools.

Chapter 13. Performing MQ-ASYNC propagation

This chapter describes how to use the main components of IMS DPROP MQ-ASYNC propagation, which are:

- The Capture program, the Apply program (and the Relational Update program (RUP) called by the APPLY program), run regularly:
  - The Capture program is called by the IMS Data Capture facility when segments are updated. If the segment is to be propagated, Capture writes changed data to an MQSeries queue
  - The Apply program monitors an MQSeries queue. Whenever a changed data message arrives on the queue, it extracts the message and passes it to the RUP
  - The RUP uses Propagation Request (PR) mapping definitions to create the SQL statements that update the target DB2 tables
- Other components that are run as required to set up the environment for the Capture and Apply

Concept of propagation requests, PRSTREAM and Apply program

PR and PRSTREAM

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.

Using the Capture component

This section describes how to use Capture to write changed data messages to the MQSeries queues.

The following sections discuss Capture in relation to:

- Input and output
- Processing
- User interaction
- Output messages
- Failure and recovery
- Return codes and error conditions
- Recovering from a failure

Input and output

Figure 41 on page 162 illustrates the sources of input to Capture and the outputs created by Capture.

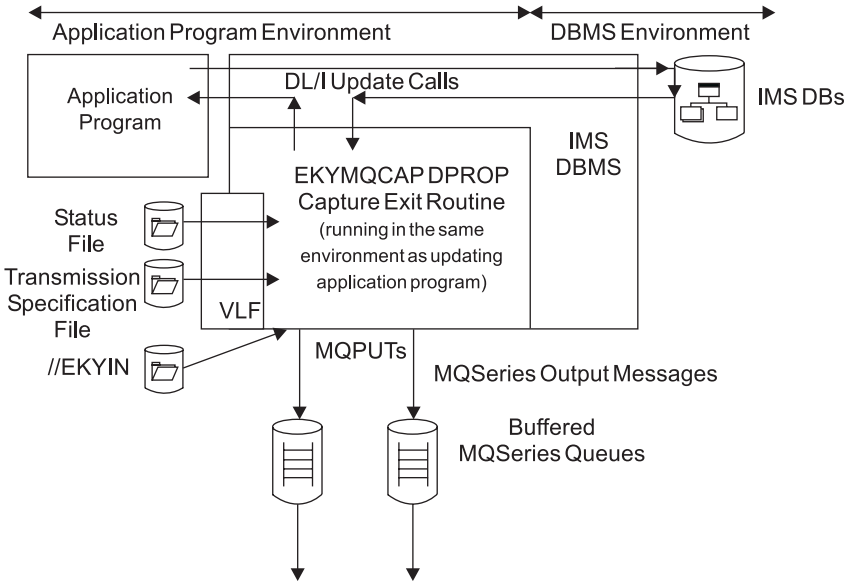


Figure 41. Capture inputs and outputs

The following list describes the input to Capture:

Input	Description
EKYTRANS	The Transmission Specification file, which is a sequential data set, contains control information such as details of propagation streams, databases, segments, fields, and queue names required to operate the Capture component.
EKYMQST	The MQ-DPROP Status file, which is a PDS member, contains the name of the Capture system and indicates whether it is active or has been emergency stopped. It also determines which VLF class will be used and the name of the default Transmission Specification file.
EKYIN	This is a sequential data set which provides a secondary source of control information for Capture. It can be used to override default options of the Capture component.

The following list describes the output from Capture:

Output	Description
MQSeries Queues	The messages placed on MQSeries queues are the primary output from the Capture component.
EKYPRINT	All errors and significant events are written to the EKYPRINT data set.
EKYWTO	Contains the text of the Write-To-Operator (WTO) messages generated by the Capture system. This is an optional data set.
EKYSNAP	Contains OS/390 snaps of the Capture task generate if Capture encounters a severe problem. This is an optional data set.
EKYTRACE	Contains information used for debugging.

Capture processing

This section describes how to use the Capture system to collect the IMS updates, package the updates as propagation streams, and send the update messages to the target. The target can be one or more Apply programs on a local or a remote site.

The Capture system:

1. Reads the contents of EKYTRANS and Status file (EKYMQST)
2. Determines which propagation streams to use.
3. Writes MQSeries messages containing update information to the appropriate queues.
4. Makes the messages available for processing or transmission when the application program commits the updates.
5. Discards the update messages from the MQSeries queue buffers if the IMS application program fails or rolls back the changes.

Reading the Capture control blocks

The Capture system:

1. The first time that it is called by the IMS Data Capture facility, the Capture exit, EKYMQCAP, reads the Transmission Specification file and builds control blocks based on the definitions of the propagation streams that you specified in the EKYTRANS file.

   Related Reading: For the syntax and description of the EKYTRANS control statements, refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210.
2. On subsequent calls, EKYMQCAP will refer to the control blocks in virtual storage to determine how to handle the updates.
3. EKYMQCAP will check periodically to determine whether the transmission specifications have been changed. If so, it will rebuild the control blocks.

Determining which propagation streams to use

EKYMQCAP reads the Capture control blocks and identifies all of the MQSeries queues that are defined for the database segment that has been updated. A database or segment may be defined for multiple propagation streams since it may be desirable to route the same update to several different targets. Each propagation stream is associated with one MQSeries queue.

A propagation stream can handle updates for several different IMS databases. The update messages for all of the databases that are defined as part of the same propagation stream are routed through the same MQSeries queue.

Writing the MQSeries messages

When IMS Data Capture calls EKYMQCAP, the information about a database update is passed to EKYMQCAP. If the transmission specification indicates that the update is to be propagated through one or more propagation streams, then EKYMQCAP builds a message consisting of a standard propagation message header and the information provided by the IMS Data Capture facility. The changed data information can include parent data as well as the changed data in the segment that was updated and the concatenated key of the updated segment.

EKYMQCAP issues a MQPUT call to place the message on each of the queues corresponding to a propagation streams which have been defined to handle the updated segment.

Making the message available for processing

EKYMQCAP invokes the MQPUT interface to write MQSeries messages that contain the IMS updates. These messages are held in the queue buffers until the two-phase commit process completes. When the IMS application program goes through the commit process, one of the resources that is committed is the MQSeries message.

After the commit is complete, the message on the queue is made available for:

- Processing by an Apply program on the same MVS image as the Capture component
- Transmission by MQSeries transmission facilities to another queue on another MVS image where the target database is located.

Discard the message on abend or rollback

If the IMS application program abends or issues a rollback call, then the MQSeries messages that have been created during the current unit of work are discarded from the queue buffers. This ensures that only committed IMS changes are propagated through the Capture system.

User interaction

You can update the Transmission Specification file (EKYTRANS) which is the main input to the MQ-DProp Status file which determines the active or inactive state of the Capture system. For full details, refer to the control statements for EKYTRANS and the description of the Capture System utility (CUT) in the *IMS DataPropagator for z/OS Reference*, SC27-1210.

Capture output messages

For details on the messages issued by the Capture system, refer to *IMS DataPropagator for z/OS Messages and Codes*, GC27-1213.

Capture failure and recovery

Because the transmission of messages is asynchronous with the updating IMS applications and MQSeries allows the definition of multiple large (4 gigabytes) Message Queue data sets, the Capture system is not affected by the unavailability of teleprocessing links, Apply programs, or target systems and can continue to operate normally.

If one of the MQSeries resources that is required by the Capture component is not available when the application updates the source database, then Capture issues an IMS user abend. This causes the updates made by the application to be rolled back and any input messages to be replaced on the IMS input queue for later processing.

Related Reading: For details on the cause of the failure and how to resolve it, refer to *IMS DataPropagator for z/OS Messages and Codes*, GC27-1213 and *IMS DataPropagator for z/OS Diagnosis*, GC27-1209.

Capture abend codes and error conditions

The abend codes and error conditions for EKYMQCAP are described in *IMS DataPropagator for z/OS Messages and Codes*, GC27-1213 and *IMS DataPropagator for z/OS Reference*, SC27-1210.

Recovering from a Capture failure

If Capture fails, examine the abend code and any messages that have been issued. Resolve the problem as described in *IMS DataPropagator for z/OS Messages and Codes*, GC27-1213 and *IMS DataPropagator for z/OS Reference*, SC27-1210. When the problem has been resolved, restart the failed IMS application programs. Capture will resume propagating the updates produced by the applications.

If it is determined that it is more important to continue running the IMS update applications than it is to propagate the changes, you can end propagation by:

- Using the CUT to "emergency-stop" the Capture system.
- Providing a “PROP OFF” control statement in the EKYIN file of a job step that updates the IMS databases.

Using the IMS Apply program

The IMS Apply program retrieves the propagation changed data messages that are written to the MQSeries queue by the Capture program and uses them to update the IMS target databases.

The following sections discuss the IMS Apply program in relation to:

- Input and output
- Processing
- User interaction
- Output messages
- Return codes and error conditions
- Special recovery and restart cases

IMS Apply input and output

Figure 42 shows the inputs and outputs from the IMS Apply program. The IMS Apply program runs as an IMS BMP job.

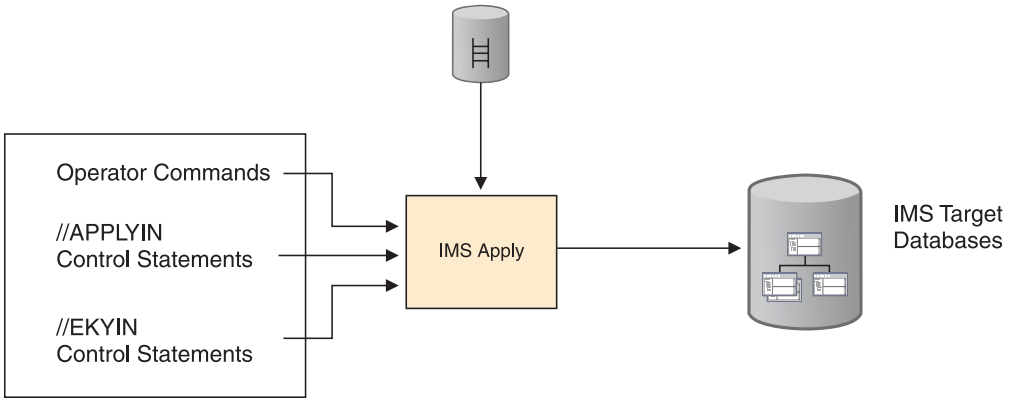


Figure 42. Inputs and outputs from the IMS Apply program

A sample JCL job is provided to run the Apply program and is described in detail in the *IMS DataPropagator for z/OS Reference*, SC27-1210. The following information is not referenced in the JCL, but also serves as input to or output from the Apply program:

MQSeries queue
Contains the records to be propagated to the IMS target databases. Written by the Capture program and read by the Apply program. Each MQSeries

queue and Propagation Stream can only be processed by a single instance of the Apply program. Two Apply programs running simultaneously cannot read from the same input queue.

IMS target databases
Specified by means of the PSB.

Related Reading: For details on each IMS Apply program input and output, refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210.

IMS Apply processing

The IMS Apply program:

- Reads the contents of the control statement file (APPLYIN)

  The APPLYIN contains the following information that you specify:

  – IMS Apply program name.
  – The number of IMS database target updates to process before a commit is issued.
  – The name of the MQSeries queue manager.
  – The name of the MQSeries queue to be processed by this instance of the Apply program.
  – The stop criteria, which determines when the IMS Apply is to stop applying updates to the IMS target databases.
  – Optionally, values to override the default setting of the error handling logic of the IMS Apply.
  – Optionally, values to override the default options for the periodic writing of performance statistics.
  – Optionally, a request that the before-image of a segment sent by the IMS source system matches the segment data in the target database segment that is to be replaced or deleted. The IMS Apply program should verify this request.
  – Optionally, identify segment types that do not have a unique key but do have multiple occurrences under their parent.
- Reads the control statements to obtain the name of the MQSeries queue that is to be used as input.
- Uses the contents of the MQSeries messages to update the target IMS databases.
- Commits the IMS target database updates, as specified in the COMMIT statement in the APPLYIN data set.
- Removes the input messages that have been processed since the last commit point from the MQSeries queue.
- Produces periodic statistics.

For further information on the APPLYIN control statements, refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210.

IMS Apply user interaction

The APPLYIN data set is the only IMS Apply program input that you can control directly. Use APPLYIN control statements to specify values for the following parameters:

APPLY NAME
The name of the IMS Apply program instance to be run. The name must be unique across all occurrences of the IMS Apply program.

COMMIT
The maximum number of MQSeries messages that can be applied before a checkpoint is issued for the IMS target database. The default value is 500.

QMANAGER
The name of the MQSeries queue manager which will be used to read the messages containing the changed data sent by the Capture system.

QUEUE
The name of the MQSeries queue that will be used as input by this instance of the IMS Apply program.

STOPAT
Indicates when the IMS Apply program is to stop processing messages. You can specify:

ID Instructs the IMS Apply program to stop processing when it reads an MQSeries message containing an event marker.

TS Indicates the IMS Apply program to process all MQSeries messages until it encounters a message that was created after the specified timestamp.

FAILURES
Provides a means to override the default setting for the error handling logic of the IMS Apply program.

NONUNIQUE
Identifies segment types that do not have a unique key but do have multiple occurrences under their parent.

VERIFY
Requests that when propagating replaces and deletes, the IMS Apply program verifies that the data of the Before image, which was sent by the source system, matches the data which is located in the segment of the target database that will be replaced or deleted.

IMS Apply output messages

All messages issued by the IMS Apply program are described in *IMS DataPropagator for z/OS Messages and Codes*, GC27-1213.

IMS Apply return codes and error conditions.

All return and reason codes shown from the IMS Apply program are described in *IMS DataPropagator for z/OS Messages and Codes*, GC27-1213.

IMS Apply program error handling

The IMS Apply program is responsible for checking return codes from IMS and MQSeries and handling propagation failures. In most cases, the default error handling logic is sufficient. For special situations, you can override the default error handling logic by using the **FAILURES** statement in the APPLYIN data set. For details of the specification of the **FAILURES** statement, refer to *IMS DataPropagator for z/OS Reference*, SC27-1210.

The IMS Apply program distinguishes between the following error types:

- Severe errors (such as internal errors)

- Unavailable resources
- Data errors (such as attempting to replace a segment that doesn't exist)
- Miscellaneous errors (such as invalid MQSeries messages)

The IMS Apply program implements backouts differently for different types of failures as follows:

Severe errors:
The IMS Apply program abends. Severe errors include those caused by IMS DPROP internal errors.

The behavior of the Apply program for severe errors cannot be influenced by the **FAILURE** control statement.

Unavailable resources:
By default, the IMS Apply program writes error messages and traces and then abends.

Data errors:
These errors include duplicate or missing target database segment occurrences. For these types of errors, by default, the IMS Apply program writes error messages and traces and, then, abends.

Miscellaneous errors:
These errors include reading an MQSeries message that does not have the expected IMS DPROP message format. For these types of errors, by default, the IMS Apply program writes error messages and traces and, then, abends.

Using the Apply program for IMS to DB2 propagation

The Apply program retrieves the propagation messages that are written to the MQSeries queue by the Capture program and passes them to the RUP, which uses them to update the DB2 tables.

The following sections discuss the Apply program 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

Apply input and output

Figure 43 on page 169 shows the inputs to and outputs from the Apply program.

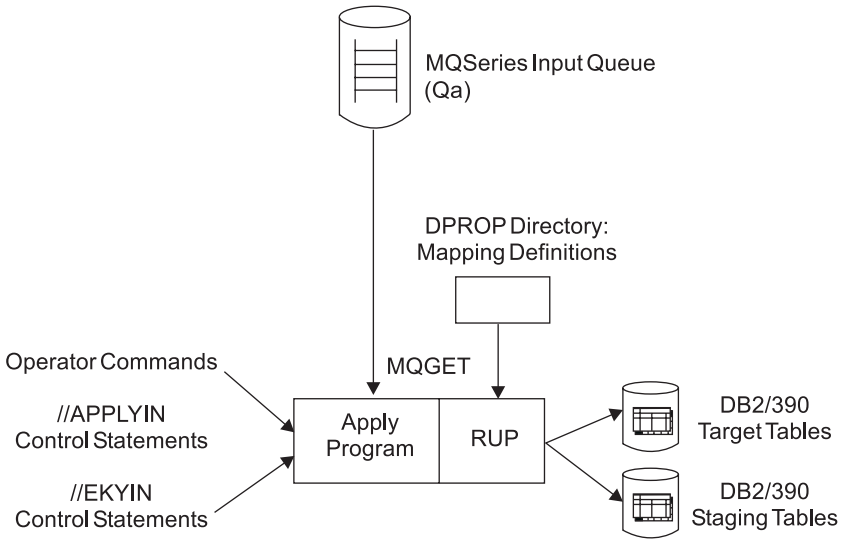


Figure 43. Apply inputs and outputs

The Apply program for IMS-to-DB2 propagation runs as an MVS batch job.

A sample JCL job is provided to run the Apply program and is described in detail in the *IMS DataPropagator for z/OS Reference*, SC27-1210. The following information is not referenced in the JCL, but also serves as input to or output from the Apply program:

MQSeries Queue
Contains the records to be propagated to the DB2 tables. Written by the Capture program and read by the Apply program. Each MQSeries queue and Propagation Stream can only be processed by a single instance of the Apply program. Two Apply programs running simultaneously cannot read from the same input queue.

DB2 tables
Specified by the mapping definitions. Updated by the RUP.

Related Reading: For details on each Apply program input and output, refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210.

Apply processing

The Apply program:

- Reads the contents of the control statement file (APPLYIN)
  The APPLYIN contains the following information that you specify:
  - Apply program name
  - DB2 subsystem name
  - DB2 plan name
  - Number of database updates to process before a DB2 commit is issued
  - The name of the MQSeries manager
  - The name of the MQSeries queue to be processed by this instance of the Apply program.
  - The stop criteria, which determines when Apply is to stop applying updates to the DB2 tables

  - Optionally, specification of the PRs to be processed by this instance of the Apply program
  - Optionally, values to override the default setting of the error handling logic of the Apply
- Reads the control statements to obtain the name of the MQSeries queue that is to be used as input.
- Uses the contents of the MQSeries messages to update the target DB2 tables, as specified in the propagation request definitions.
- Commits the DB2 updates, as specified in the COMMIT statement in the APPLYIN data set. In order to avoid causing lengthy enqueues on DB2 resources, Apply commits DB2 updates every few seconds even if the number of messages specified on the COMMIT statement has not been reached.
- Removes the input messages that have been processed since the last commit point from the queue.

For further information on the PRCT and the APPLYIN control statements, refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210.

Apply user interaction

The APPLYIN data set is the only Apply program input that you can control. Use APPLYIN control statements to specify values for the following parameters:

APPLY NAME
The name of the Apply program instance to be executed. The name must be unique across all occurrences of the Apply program.

DB2 SYSTEM
The MVS subsystem name of the DB2 system to be accessed.

PLAN
The name of the DB2 plan to be used by the Apply program.

COMMIT
The maximum number of MQSeries messages that can be applied before a DB2 commit is issued. The default is **500**.

QMANAGER
The name of the MQSeries Queue Manager which will be used to read the messages containing the changed data sent by the Capture system.

QUEUE
The name of the MQSeries queue that will be used as input by this instance of the Apply program.

STOPAT
Indicates when the Apply program is to stop processing messages. You can specify:

ID Instructs the Apply program to stop processing when it reads a message containing an Event Marker with the specified identifier. This is the default.

TS Indicates that the Apply program should process all messages until it encounters a message that was created after the specified timestamp.

INCLUDE Indicates that the Apply program should process only the PRs that are specified on the INCLUDE statement. The default is

that the Apply program will process all PRs that apply to the updated segment in an input message.

EXCLUDE Indicates that the Apply program should process all PRs that apply to the updated segment in an input message except for those PRs that are specified on the EXCLUDE statement.

FAILURES Provides a means to override the default setting for the error handling logic of the Apply program.

Apply output messages

All messages issued by the Apply program are described in *IMS DataPropagator for z/OS Messages and Codes*, GC27-1213.

Apply return codes and error conditions

The return and reason codes issued from the Apply program are described in *IMS DataPropagator for z/OS Messages and Codes*, GC27-1213.

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 Apply program 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 Apply program.

IGNORE The RUP

- Reports the propagation request failure to //EKYPRINT.
- Passes a return code of 0 and a reason code of 0 to the Apply, as though no failure has occurred. The Apply program is not aware that an error has occurred and continues normally.

IGNORE is usually used only for testing propagation requests.

Apply program error handling

The Apply program is responsible for checking the return codes from the RUP and handling propagation failures. In most cases, the default error handling logic will be sufficient. For special situations, the default error handling logic can be overridden by using the FAILURES statement in the APPLYIN data set. For details of the specification of the FAILURES statement, refer to *IMS DataPropagator for z/OS Reference*, SC27-1210.

The default error handling logic of the Apply program is:

- Deadlock and Time-out

  The Apply program will retry the deadlocked or timed-out Unit-of-Work without any loss of data or abend. specifying:
- Mapping errors

The Apply program will handle the error according to the ERROPT specified for the failing PR.

– If the ERROPT is not IGNORE

  The Apply program will write error messages and traces and will then abend. After the problem indicated by the error messages has been resolved, the Apply program can be restarted without losing any data.

– If the ERROPT is IGNORE

  The Apply program will:

  - Write error messages
  - Write the database update causing the problem to an error table
  - Skip further processing of the database updates that caused the problem.

• Other errors

  All other error types are handled in the same way as mapping errors where the ERROPT is not IGNORE.

Chapter 14. Propagating IMS data to staging tables

You can use MQ-ASYNC propagation 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 DB2 DataPropagator, an IBM Copy Management tool that can copy relational data from DB2 for z/OS to many other relational databases. For more information about DB2 DataPropagator, see the*IBM DB2 UDB Replication Guide and Reference*, SC27-1121.

The staging tables are referred to as Consistent Change Data (CCD) tables.

Figure 44 shows how you can propagate operational data from IMS to DB2 for z/OS and DB2 DataPropagator.

Propagator IMS Data to DB2 and DB2 for OS/2

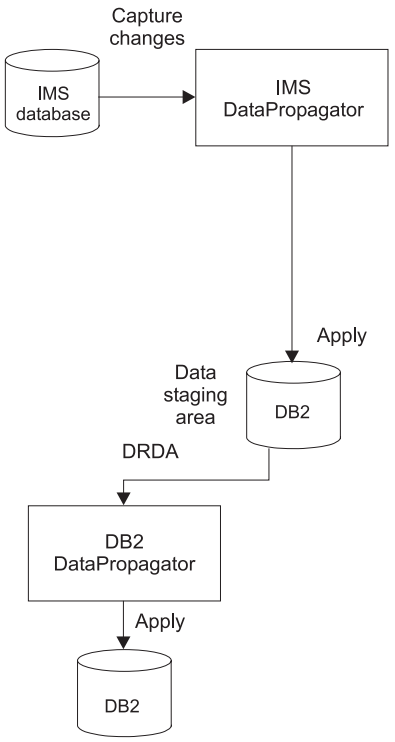


Figure 44. Propagating IMS data to DB2 DataPropagator

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
- Setting the values of technical columns of the 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.

Note: This always contains a zero when updated by the MQ Apply program.

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.

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 177).

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.

Related Reading: DB2 DataPropagator also offers facilities for pruning. For details, see *IBM DB2 UDB Replication Guide and Reference*, SC27-1121.

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 DB2 DataPropagator 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 175.

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 175. 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 177.

Setting the technical columns of the staging tables

This section describes how the Apply program will set the values of technical columns of the staging tables.

IBMSNAP_COMMITSEQ
This DB2 Column, which is defined as ’CHAR(10) FOR BIT DATA NOT NULL’, will be set by Apply to the 8-byte store clock value of the IMS database change on the source system and will be padded with hexadecimal zeroes, followed by a trailing bit set to 1. The trailing 1 is to avoid inadvertent use by LOG-ASYNC Receiver and MQ-ASYNC Apply of the same COMMITSEQ value for updates that are not related.

Notes:
- LOG-ASYNC uses the STCK value of the IMS Commit, as stored in the IMS Commit log record.
- For this column, the values stored by MQ-ASYNC Apply and LOG-ASYNC Receiver are similar and will not cause a problem for conversion purposes (LOG-ASYNC to MQ-ASYNC), assuming the use of a database quiesce point for the migration/fallback.
- Notice, however, that if both MQ-ASYNC and LOG-ASYNC store into staging tables updates of the same IMS UOW, a program reading the staging tables will not be able to recognize that rows inserted by MQ-ASYNC belong to the same UOW as rows inserted by LOG-ASYNC, because of the different trailing bit.

IBMSNAP_INTENTSEQ
Will be set to 0 by MQ-ASYNC.

IBMSNAP_LOGMARKER
This DB2 timestamp column will be set to the local time of the IMS database change on the source system.

Notes:
- This column optionally provides an indication of the currency of the update at the source database.
- LOG-ASYNC sets it to the source (IMS) commit time.
- For this column, MQ-ASYNC and LOG-ASYNC are sufficiently consistent for conversion purposes (assuming use of a database quiesce point for the conversion/fallback).

A program reading changes in the CCD tables propagated by MQ-ASYNC, will have the impression that every IMS change has been done in its own IMS Unit Of Work.

The ASN.IBMSNAP_REGISTER table

After a CCD is registered in the ASN.IBMSNAP_REGISTER table, as the MQ Apply program continues to apply updates to the CCD in that table, the program also updates the following:

- SYNCPOINT column with the Commitseq from latest CCD row update
- SYNCTIME column with the Logmarker from latest CCD row update

Chapter 15. MQ-ASYNC considerations

You have flexibility available when implementing and managing an IMS DPROP MQ-ASYNC system. This chapter provides two Apply user scenarios and outlines some of the considerations that must be taken into account in the areas of:

- Remote site considerations
- Recommendations for database administration with IMS DPROP MQ-ASYNC
- Synchronization scenario with an Extract/Load
- Error recovery
- Event Marker Facility (EMF)
- Daylight saving time change considerations

User operations scenarios

IMS DPROP MQ-ASYNC Apply programs support two user operations scenarios:

- Near-real-time propagation

  In this scenario, the Apply program occurrences are permanently active.
- Point-in-time propagation

  In this scenario, the Apply program occurrences are started at the time chosen for updating the target table or database. The program occurrences can be stopped automatically when processing the Event Marker MQSeries message which identifies the source system point-in-time that must be reflected in the content of the targets.

An Apply program for IMS-to-DB2 propagation runs as a long-running non-IMS job, an IMS BMP job, or as an IMS batch job. An IMS Apply program **must** run as an IMS BMP job.

Multiple Apply program occurrences can run in parallel:

- For more throughput
- For more operational flexibility
- With no time-coordination of target updates across different Apply jobs
- With the following possible granularities:
  - One Apply job per target DB2 Table
  - One Apply job per target IMS database
  - One Apply job per keyrange of IMS root segment
  - One Apply job per target DB2 table and keyrange of IMS root
  - One Apply job per target IMS database and keyrange of IMS root

Note that the same MQSeries queue cannot be processed concurrently by more than one Apply program occurrence.

Remote site considerations

The following IMS DPROP asynchronous MQSeries 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.

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 subsections 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.

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 for database administration with IMS DPROP MQ-ASYNC

A PRSTREAM 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 PRSTREAMs, the source UOW is also split. UOW consistency cannot be maintained across multiple DB2 apply programs.

If PRSTREAMs for related tables are split between multiple APPLY program occurrences, 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 APPLY program occurrences. If the APPLY program occurrence that 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, you can avoid such problems.

Although executing multiple APPLY program occurrences 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.

Synchronization scenario with an Extract/Load

Before starting to propagate IMS data changes to a specific table or to a specific group of DB2 tables, the Database Administrators (DBAs) should populate the DB2 copy using IBM DataRefresher (or another extract program). This process is referred to as the (initial) synchronization of the DB2 copy with the IMS copy[12].

A (re-)synchronization of the DB2 copy is also necessary after a LOAD of the IMS database changes the content of the propagated IMS data.

Synchronization with an Extract/Load can also be used to reinstate the consistency of the two database copies after the DB2 copy of the data has become inconsistent with the IMS copy. For example, because of severe operational or propagation errors such as a cold start of MQSeries, or after a timestamp recovery of the IMS database.

From an operational point of view, what does a typical synchronization scenario look like?

1. First the IMS database must be quiesced to be able to extract the data with integrity. IMS full-function databases can be quiesced by using the READONLY command of the DPROP SCU Utility. The DPROP SCU will issue a message showing the quiesce timestamp. To quiesce a DEDB, the DEDB must be taken offline.
2. In situations when it is important to stop the Apply program after it processes the last IMS database change performed before the quiesce point, the MQ-ASYNC Event Marker is used on the source system to create an Event Marker MQSeries message.

   Note: STOPAT control statements of the Apply program should request an automatic stop as soon as the Apply program receives this message. Because MQSeries messages are transmitted FIFO, this is after the processing by the Apply of the IMS database changes performed before the quiesce point, and before the processing by the Apply of IMS database changes performed after the quiesce point.
3. Then the extract process is run against the IMS database, while it is quiesced. When the extract is complete, the IMS database can be made available for updates (for full-function databases, use the READOFF SCU command; for DEDBs, take the database online).

   A possible alternative:
   - IMS standard recovery utilities are used to create a shadow IMS database copy that is recovered with a timestamp recovery to the quiesce point.
   - The shadow database copy is then used as input to the extract.

   In this alternative, the main copy of the IMS database can be made available for updates much faster than a scenario without a shadow copy; the database can be made available for updates as soon as both its quiesce and the Event Marker process have successfully completed.
4. The file containing the extracted data is transmitted to the target system.

12. Note however, that if you are propagating IMS data to a staging table that does not have the Complete attribute, an initial synchronization is not necessary.

Typically, when operating in near-real-time, the Apply processing has already reached the database quiesce point, and the Apply has already stopped after reading the Event Marker message. If this is not the case, wait until the Apply processing reaches the database quiesce point and stops.

The extracted data is loaded into the DB2 copy. At the completion of the load, do not forget to save the loaded DB2 tables for eventual recovery purposes.

5. The Apply program is started or restarted.

   Optionally, depending on the situation (when the MQSeries queue contain IMS updates that are older then the quiesce timestamp and should not be applied to the freshly loaded DB2 target table), you can use an EXCLUDE UNTIL= control statement on the Apply program to skip the apply/propagation process of the IMS database changes that have been done before the quiesce timestamp. To specify the correct quiesce timestamp on UNTIL=, the DBA:
   - Can use the timestamp displayed by the SCU READON command in item 1 above for full-function databases
   - For DEDBs, the DBA must determine when the DEDB has been taken offline.

   Note that skipping the apply/propagation process can be granular at the level of the PR, PRSET, DBD, or segment-type.

The management of changes and recovery that affect the source or target data for propagation will be a significant responsibility for the administrator of IMS DPROP asynchronous MQSeries propagation. Just as you need initial synchronization when first implementing asynchronous MQSeries propagation, you need synchronization when recovering from failures and making changes to mapped data.

Event Marker Facility (EMF)

See “The MQ-ASYNC Event Marker facility” on page 21 for a description for Event Marker facility.

Following are considerations when using the Event Marker facility:

- To identify to IMS DPROP MQ-ASYNC a source system event (for example, the logical end of a business day or completion of a monthly job streams), run the Capture System utility (CUT) on the source system when the event happens. With CUT control statements, you can assign an Event Marker ID to the event.

  For example, on the source system, run the CUT at the desired point-in-time with the following control statement:

  ```
  EM  ID=event-marker-id, PRSTREAM=name-of-prstream(s)
  ```

- The CUT will create an MQSeries message that will be transmitted in FIFO sequence to the Apply with all other MQSeries messages of IMS DPROP MQ-ASYNC.
- The Apply Program can be instructed by means of control statements to stop its processing when reading an MQSeries message containing an Event Marker with a specific Event Marker ID. After it stops, the content of the target tables will reflect the content of the source IMS databases at the point-in-time of the identified event.

  For example, on the target system, provide in the //APPLYIN file of Apply program the following control statement:

  ```
  STOPAT ID=event-marker-id
  ```

- MQ-ASYNC also supports a callable interface for the creation of Event Marker message on the source system. This allows the creation of Event Marker

messages from within programs. For details of the callable interface, see the *IMS DataPropagator for z/OS Customization Guide*, SC27-1214.

Daylight Saving Time change considerations

A Daylight Saving Time change occurs when clocks are seasonally adjusted in Spring and in Fall.

The time change in Spring that advances the clock by one hour is not problematic for MQ-ASYNC.

For MQ-ASYNC, the problematic time change is in the Fall when the clock of the source systems are set back by one hour. In terms of source system timestamps, this results in duplicate points-in-time that have identical timestamp values. This results in the following minor impacts on MQ-ASYNC:

1. Use of the STOPAT TS= control statements of the Apply program that specifies a source system timestamp within the 2-hour interval around the Daylight Saving Time change will be affected.

   The Apply can stop its processing at the wrong source system point-in-time. However, after its restart, the Apply program will continue its message processing where it previously stopped. No IMS changes will be lost or applied twice.

   Recommendation: Do not use the STOPAT TS= control statement specifying a source system timestamp within this interval. Consider using a STOPAT ID= control statement instead.
2. The rarely used EXCLUDE UNTIL= control statements of the Apply program will be affected if they specify on the UNTIL= keyword a source system timestamp within the 2-hour interval around the Daylight Saving time change.

   The Apply can exclude from its processing (or include in its processing) the wrong IMS changes.

   Recommendation: Plan the rare activities that require use of EXCLUDE UNTIL= so that you do not need to specify a timestamp within the 2-hour interval on the UNTIL= keyword.

Chapter 16. Verifying consistency between IMS and DB2 data copies (CCU)

This chapter is an overview of the IMS DPROP Consistency Check utility (CCU), which checks the consistency of propagated data between IMS source and DB2 targets and generates repair statements if errors are found.

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 the RUP 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 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 45 on page 188. If the CCU finds inconsistencies, it generates SQL statements to correct them. You can run the SQL statements through DSNTEP2 or DSNTIAD to restore DB2 tables to the same data content as the IMS database.

CCU Execution and Repair Process

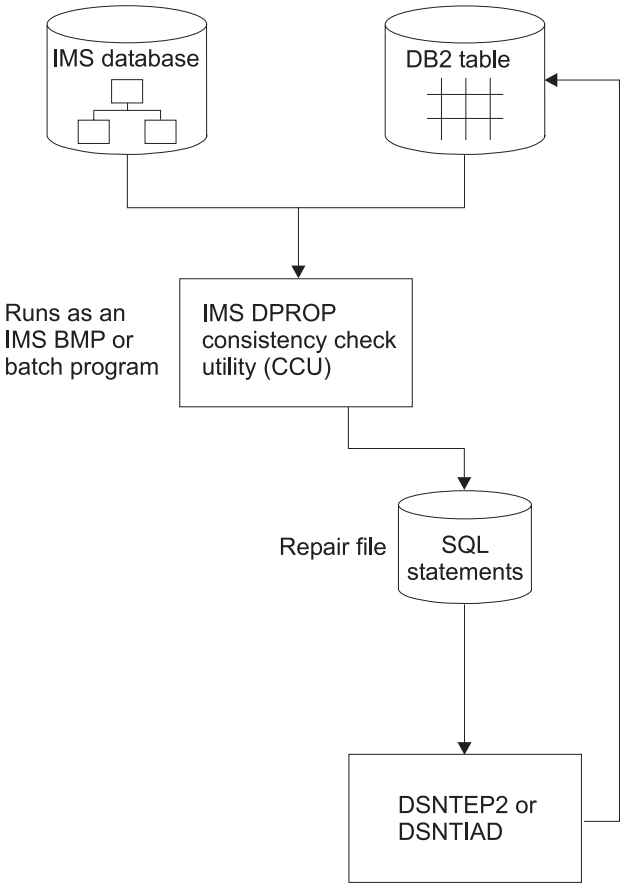


Figure 45. CCU Execution and the Repair Process

The following sections discuss:

- When to use the CCU
- Considerations for MQ-ASYNC 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 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 MQ-ASYNC propagation

In MQ-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 you intervening. 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 Apply program using the quiesce point as a stop time. If you keep the IMS database in read-only state and refrain from further DB2 table updates after all the updates are applied up to the IMS quiesce point, then you can run the CCU against the actual IMS database and the DB2 copy.

Alternately, the CCU can use an IMS image copy as a source and this allows IMS updates to be resumed after the image copy is made.

You can also 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.

Considerations when concurrent updates are being done

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 191 and the *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*, SC27-1210. 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 191.

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 multi-step 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.

If the inconsistency is in category

1 The CCU generates an SQL INSERT statement for the missing DB2 row

2 The CCU generates an SQL DELETE statement

3 The CCU generates an SQL UPDATE statement for the inconsistent columns

For every data inconsistency, you must determine whether you want to make data consistent by applying the generated SQL statement to the DB2 tables using the DSNTEP2 or DSNTIAD programs of DB2.

The CCU writes the generated repair statements to sequential files that you can edit with TSO/ISPF before passing the files to DSNTEP2, DSNTIAD, 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*, SC27-1210.

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 rather than reestablish consistency with CCU-generated repair statements.

To verify your propagation request, rerun the CCU after reloading the data.

Some 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.
- 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
- For MQ-ASYNC propagation, updates are not propagated before the CCU runs

There can be other causes for inconsistencies. Determine the cause and eliminate the problem if the inconsistencies are critical.

Chapter 17. Problem determination tools

This chapter explains how to get information about various database objects and system activities using IMS DPROP's auditing and tracing facilities, message table, and CCU. The diagnostic tools provided by IMS DPROP can help you identify and resolve problems.

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.

Start the IMS DPROP trace by putting a TRACE control statement in the //EKYIN file allocated to the job step in which IMS DPROP is executed.

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 MQSeries 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's EKYZ620X exit routine of the IMS File Select and Formatting Print utility (DFSERA10).

The *IMS DataPropagator for z/OS Diagnosis*, GC27-1209, contains detailed information on IMS DPROP trace functions.

IMS DPROP 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

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 IMS DPROP's SMF records, update the SMFPRMxx member of SYS1.PARMLIB at your installation. Additional information on this subject is available in the *IMS DataPropagator for z/OS Installation Guide*, GC27-1212.

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; the format of the audit trail table is described in the *IMS DataPropagator for z/OS Reference*, SC27-1210.

The audit trail shows:
- When the CCU last ran
- When the data was last known to be consistent
- When the RUP 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.

Additional information, sample JCL, and control statements for AUDU and the Audit Trail table are described in *Reference*.

The Figure 46 summarizes the audit trail generation process.

IMS DPROP Audit Process

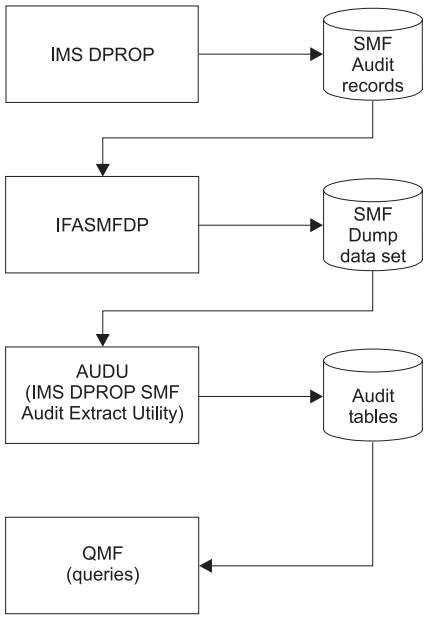


Figure 46. Overview of the IMS DPROP audit process

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 SMFPRMxx 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 to 7 do not apply to IMS DPROP asynchronous MQSeries Selector-only sites that do not have DB2 installed. See *IMS DataPropagator for z/OS Reference*, SC27-1210 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. Users who SELECT,

UPDATE, INSERT, DELETE, and maintain the table (for example, deleting old records) must be granted update privileges.

For users loading the tables with the AUDU, you should grant EXECUTE privilege on the plan of the AUDU utility.

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 16, “Verifying consistency between IMS and DB2 data copies (CCU),” on page 187 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 IMS DPROP asynchronous MQSeries Selector-only sites.

Related Reading: Refer to the *IMS DataPropagator for z/OS Reference*, SC27-1210 for more information on the message table.

Chapter 18. Performance and monitoring

Unlike IMS, DB2 and MQSeries, IMS DPROP has few components that you can tune. Factors that allow IMS and DB2 to perform at their optimum levels help the performance of MQ-ASYNC as well. 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.

With MQ-ASYNC DPROP, the Capture component, the MQSeries, and the APPLY component execute asynchronously to each other. Each one can execute at its own maximum speed, without being slowed down by the other two.

MQ-ASYNC exploits high performance messaging services of MQSeries. To tune the performance of MQSeries, refer to MQSeries documentation. Some of factors that can be considered:

- MQ logging
- MQ queue buffers
- Transmission specific parameters in MQSeries, like the number of messages per Transmission UOW and VTAM and TCP/IP parameters (such as RU size and compression)
- Dispatching priority of MQ queue manager

To allow high propagation rate, fast DASD device that has few other activities should be used for MQ logging.

MQ-ASYNC propagation supports:

- Multiple updates per message
- Compression
- The execution of multiple Apply program occurrences in parallel with multiple PR streams
- The tuning of the Commit frequency of the Apply program

This chapter discusses:

- MQ-ASYNC performance
- Monitoring propagation

IMS DPROP MQ-ASYNC 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 for IMS-to-DB2 propagation
- Setup phase
- Propagation phase for IMS DPROP
- CCU execution

Mapping and design phase for IMS-to-DB2 propagation

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.

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.

With IMS-to-DB2 propagation, 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.

With IMS-to-IMS propagation, the method for extracting data from an IMS database and loading an IMS target database can take considerable time. The elapsed time can be influenced by extract and load method you select. For example, using the Image Copy utility to obtain a copy of the IMS source database and then using the IMS Recovery utility with the source Image Copy will probably be faster than using the HD Unload utility and HD Reload utility for performing extracts and loads.

Propagation phase

The performance of IMS DPROP MQ-ASYNC can be greatly influenced by:

- Performance characteristics of MQSeries, like the transport of the changed data from the source system to the target system.
- Apply by RUP of the changes to the target DB2 tables

CCU Execution for IMS-to-DB2 propagation

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. For more information on the IMS Monitor, refer to *IMS/ESA Administration Guide: System*, SC26-8013 and *IMS/ESA Administration Guide: Database Manager*, SC26-8012. Information on how to run the DB2 Performance Monitor is in *DB2 PM Report Reference for OS/390*, SC27-0853.

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.

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.

For more information about the IMS Database Tools, see http://www-306.ibm.com/software/data/db2imstools/library.html.

Part 5. Appendixes

Notices

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

IBM Corporation
J64/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:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © 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

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AD/Cycle
AT CICS
CICS/ESA
CICS/MVS
COBOL/370
Database 2
DataPropagator
DataRefresher
DB2
DFSMS
DXT
IBM
IMS
IMS Client Server/2
IMS/ESA
Information Warehouse
Language Environment
MVS
MVS/ESA
OS/390
QMF
RACF
SAA
z/OS

Adobe, Acrobat, Portable Document Format (PDF), and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

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 which reads the MQSeries messages originally created by EKYMQCAP containing the IMS changed data and passes it to the RUP. RUP transforms the changed data into relational format and updates the DB2 target tables. With the addition of the new IMS Apply program, the term Apply program may also be used to describe a generic program when it applies to both the program that initiates DB2 updates and IMS target updates. When it is appropriate to explicitly distinguish between the two types of Apply programs (in order to avoid ambiguity), the Apply program for IMS may be further identified as the "IMS Apply program".

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 by means of 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 IMS databases or DB2 tables of this occurrence of the Apply program reflect the content of the IMS source databases at the time that the Event Marker ran on the source system.

The Event Marker is used for an automated stop of the Apply Program when the content of the target IMS databases or 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 Apply program. A component of IMS MQ DPROP. This component reads the MQSeries messages originally created by EKYMQCAP which contain the IMS changed data and uses the data to update the IMS target databases.

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 by means of
| MQSeries messages to an IMS DPROP Apply program.
| With IMS-to-IMS replication, the IMS Apply program
| reads the MQSeries messages and applies the new
| data to the IMS database targets. With IMS-to-DB2

replication, the Apply program reads the MQSeries messages, transforms the data into relational format, and then applies the new data to the target DB2 tables.

MQ-ASYNC supports both near-real time propagation and automated point-in-time propagation.

MQSeries. A family of IBM licensed programs that provide message queuing services.

MQSeries for OS/390. The members of the MQSeries that run on OS/390 systems.

MSDB. Main storage database.

MSC. Multisystem communication.

MVG. Mapping Verification and Generation.

MVG input tables. A group of DB2 tables into which the DBA stores propagation request definitions when DataRefresher is not used to provide mapping information. Once the propagation requests are stored, the DBA invokes MVGU. MVGU invokes MVG, that validates the propagation request and copies the mapping definitions from the MVG input tables to the IMS DPROP directory.

MVGU. Mapping Verification and Generation utility.

N

Near RealTime. A delay of only a couple of seconds.

O

OLDS. Online Data Set.

One-way DB2-to-IMS propagation. This is the propagation of changed DB2 tables to IMS segments. Distinguish between:
- One-way DB2-to-IMS propagation
- DB2-to-IMS propagation, as part of two-way propagation

One-way IMS-to-DB2 propagation. This is the propagation of changed IMS segments to DB2 tables. Distinguish between:
- One-way IMS-to-DB2 propagation
- IMS-to-DB2 propagation, as part of two-way propagation

P

PCB. Program communication block.

persistent MQSeries message. An MQSeries message that survives a restart of the MQSeries Queue Manager.

physical concatenated key. See IMS physical concatenated key.

Point In Time Propagation. An Asynchronous propagation is said to operate in 'Point In Time' mode, when the data content of the target databases matches the content of the source databases at a previous, clearly identified Point In Time. For example, a Point In Time Propagation can be used to reflect in the content of the target databases the logical end of a business day, or the logical end of business month, or the end of specific Batch jobstream that updated the source databases.

PR. Propagation request.

PR ID. Propagation request identifier.

PRCT. Propagation Request Control Table

PRDS. Propagation Request Data Set

PRDS register file. A data set created by the IMS DPROP asynchronous propagation Selector that contains details of the associated PRDS.

PRDS register table. An IMS DPROP directory table that is created at the Receiver site when IMS DPROP is installed. The table is initially empty and you must populate it, using the PRU REGISTER control statements.

PRDS Registration utility (PRU). An IMS DPROP asynchronous propagation utility that registers PRDSs in the PRDS Register Table.

propagation. See data propagation.

Propagation Data Stream. A stream of changed IMS data that flows in MQSeries messages from the Capture Component of IMS DPROP to the Apply Component of IMS DPROP. Propagation data streams are defined with PRSTREAM control statements in the //EKYTRANS file of EKYMQCAP.

propagation delay. The time elapsed between the update of the IMS source database by the application programs and the update of the target DB2 table by IMS DPROP.

Propagation exit routine. An IMS DPROP exit routine you can write to propagate data when the generalized mapping cases don't meet your needs. A Propagation exit routine must provide all the logic for data mapping, field conversion, and propagation.

propagation group. A subset of the propagation requests in the IMS DPROP directory propagation request table (IMS DPROP asynchronous only).

You can define as many propagation groups as you like, but any propagation request can be associated with one and only one propagation group.

propagation log records. IMS log records that the IMS DPROP asynchronous propagation Selector writes to PRDSs:

- 9904 (update) records
- Commit or abort records
- SETS/ROLS records

propagation request control table (PRCT). An IMS DPROP directory table that is created at the Receiver site when IMS DPROP is installed. It contains details of all propagation requests defined to IMS DPROP and, in combination with the RCT, enables the Receiver to ascertain:

- Which propagation requests are assigned to which Receivers
- The activity status of all defined Receivers
- The activity status of all propagation requests that are assigned to defined Receivers

Propagation Request data set (PRDS). A sequential file into which the IMS DPROP asynchronous propagation Selector writes all propagation log records for a propagation group.

propagation request (PR). A request to propagate data between IMS and DB2. You define propagation requests for each segment type that is to be propagated.

PR set. A group of logically related propagation requests, identified by having the same PRSET ID. PR sets are typically used when you propagate the same IMS data to multiple sets of DB2 tables.

PRU. PRDS Registration utility.

PSB. Program specification block.

R

RCT. Receiver control table.

Receiver. An IMS DPROP asynchronous propagation component that retrieves the propagation log records from a PRDS and passes them to the RUP, that uses them to update the DB2 target tables.

Applies to LOG-DPROP.

RECEIVER control statement. A control statement that is input directly into the IMS DPROP asynchronous propagation Receiver JCL to specify:

- The name of the Receiver that is to process a PRDS
- The names of the DB2 subsystem to be accessed and the DB2 plan
- The number of committed UOWs to process before a DB2 commit is issued

Applies to LOG-DPROP.

Receiver control table (RCT). An IMS DPROP directory table, that is created at the Receiver site when IMS DPROP is installed. The table is initially empty and you must populate it, using the SCU CREATEREC control statement. It contains details of all Receivers and, in combination with the PRCT, enables the Receiver to ascertain:

- Which propagation requests are assigned to which Receivers
- The activity status of all defined Receivers
- The activity status of all propagation requests that are assigned to defined Receivers

Applies to LOG-DPROP.

Relational to Hierarchical propagation. This is one-way relational to hierarchical propagation: the one-way propagation of changed DB2 tables to IMS segments. The terms *relational to hierarchical propagation* and *one-way DB2-to-IMS propagation* are interchangeable.

relational update program (RUP). The IMS DPROP component that does the actual IMS to DB2 propagation. RUP is the IMS DPROP-provided IMS Data Capture exit routine.

For synchronous propagation, the IMS Data Capture function calls RUP with the changed IMS segments.

For user asynchronous propagation, your routine gets the changes from IMS and later calls RUP.

For IMS DPROP asynchronous propagation, the Receiver gets the changes from the Selector-Receiver Interface and later calls RUP. In either case, RUP propagates the changes to DB2.

RIR. RIR is an IMS DPROP abbreviation for DB2 Referential Integrity Relationship. Database administrators can define RIRs between tables in order to request that DB2 catches and prevents update anomalies in the relational databases.

Implementation of RIRs between propagated tables is:

- Optional for one-way IMS to DB2 propagation
- Strongly recommended for DB2 to IMS and two-way propagation

RTT. Resource translation table.

RUP. Relational Update program.

RUP control block table. A single IMS DPROP directory table that contains one RUP propagation control block (PRCB) for each propagated segment type. Each RUP PRCB contains details of the relevant database and segment.

S

SCF. Selector Control File.

SCF Apply job. Uses the SCF control statements to create new propagation groups and to list and modify existing propagation groups in the SCF.

SCF Compare job. Used to compare the contents of the Group Definitions File with the propagation groups in the SCF and to generate SCF control statements to bring the SCF into line with the Group Definitions File.

SCF control statements. Can be generated automatically by the IMS DPROP asynchronous propagation GUU or input directly into the IMS DPROP asynchronous propagation SCF Apply utility JCL. The control statements modify the contents of the SCF records.

SCU. Status Change utility.

segment exit routine. An IMS DPROP exit routine you can write to complement the logic of the generalized mapping cases. Segment exit routines are typically used to convert a changed data segment from the form it has in your IMS database to a form you have defined in your propagation request.

SELECT control statements. Control statements that are input directly into the IMS DPROP asynchronous propagation Selector JCL to define the execution options for the Selector.

Applies to LOG-DPROP.

Selector. An IMS DPROP asynchronous propagation component that collects propagation log records from the IMS log files and writes them to PRDSs for later processing by the IMS DPROP asynchronous propagation Receiver component.

Applies to LOG-DPROP.

Selector control file. Created at Selector installation or generation time and contains the following control information that is essential to the operation of the Selector:

- Database records and propagation group records
- DBRC information
- Timestamp information

Applies to LOG-DPROP.

SLDS. System Log Data Set.

SNAP. system network analysis program

Source System. An OS/390 system where IMS source databases of the IMS DPROP propagation reside.

SQL update module. A module generated by MVG for each propagation request belonging to a generalized mapping case. An SQL update module contains all the SQL statements required to propagate to DB2 the changed IMS data for that propagation request.

SSM. Subsystem member. An IMS JCL parameter that identifies the PDS member that describes connection between IMS and the DB2 subsystems.

Status Change utility (SCU). An IMS DPROP utility that:

1. Changes the status of propagation requests in the synchronous environment. Propagation requests can be active, inactive, or suspended. The SCU also performs a variety of other service functions.
2. Maintains the Timestamp Marker Facility and populates the RCT and the PRCT in IMS DPROP asynchronous propagation.

synchronous propagation. The propagation of data within the same unit-of-work as the update call.

T

Target System. An OS/390 system where DB2 target tables of the IMS DPROP propagation reside.

Timestamp Marker Facility. Supports the statements that create, assign, and delete timestamp markers in the SCF. It is run as part of the SCU.

TSMF. Timestamp Marker Facility.

TSMF Callable Interface. A facility that allows a user application to create a stop timestamp for one or more propagation groups.

Two-way propagation. The combination of IMS-to-DB2 propagation and DB2-to-IMS propagation for the same data.

TW propagation. See two-way propagation.

U

UIM. User Input Manager.

ULR. Uncommitted Log Record.

uncommitted log records (ULR). When the IMS DPROP asynchronous propagation Selector terminates, it writes all uncommitted log records (propagation log records that have not yet been either committed or aborted by IMS) to the uncommitted log record data set. On a subsequent Selector execution, these records will be either written to the appropriate PRDS (if they have been committed by IMS) or deleted from the uncommitted log record data set (if they have been aborted by IMS).

UOW. Unit of work.

USER-ASYNC. The User asynchronous propagation functions of IMS DPROP.

user exit. See exit routines.

User Input Manager (UIM). A DataRefresher component to which you describe your IMS databases and the mapping between IMS databases and DB2 tables. The mapping is defined by submitting extract

requests. You can specify on an extract requests that the UIM is to invoke the DataRefresher Map Capture exit routine provided by IMS DPROP and pass it the DataRefresher mapping definitions of the extract request.

user mapping case. A mapping case you can develop if the generalized mapping cases don't meet your needs.

V

Virtual Lookaside Facility (VLF). An MVS/ESA component that is a specific implementation of data spaces. IMS DPROP exploits VLF for a high-performance retrieval of mapping information and other control information.

VLF. Virtual Lookaside Facility.

Bibliography

The IMS DataPropagator for z/OS Version 3 Release 1 Library

Order Number	Book Title
SC18-7048	Administrators Guide for Log Asynchronous Propagation
SC18-7056	Administrators Guide for MQSeries Asynchronous Propagation
SC18-7055	Administrators Guide for Synchronous Propagation
SC18-7047	Concepts
SC18-7054	Customization Guide
GC18-7053	Diagnosis
GC18-7049	An Introduction
GC28-7051	Installation Guide
G18-7050	Messages and Codes
SC18-7052	Reference
GC27-1628	Licensed Program Specification

Other Books Referenced in This Book

The following books are referred to in this book or might be helpful in understanding administration tasks:

SC26-4888 DB2 Administration Guide, Version 5

SC26-4891 DB2 Command Reference, Version 5

SC26-3395 DB2 Utility Guide and Reference, Version 5

SC26-4890 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 5

SC26-3062 IMS/ESA Application Programming: Database Manager, Version 5

SC26-3065 IMS/ESA Administration Guide: Database Manager, Version 5

SC26-3064 IMS/ESA Customization Guide, Version 5

SC26-3076 IMS/ESA Installation Volume 2: System Definition and Tailoring, Version 5

SC26-4329 IMS/ESA Utilities Reference: System, Version 5

SC26-3072 IMS/ESA Operations Guide, Version 5

GC26-8031 IMS Release Planning Guide, Version 5

SC26-3075 IMS/ESA Administration Guide: System, Version 5

SC26-4627 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

SC26-9920 IBM DB2 Universal Database Replication Guide and Reference

Index

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Program Number: 5655-E52

Printed in USA

SC27-1217-02