IBM IMS DataPropagator for z/OS

Administrators Guide for Synchronous Propagation

Version 3 Release 1

SC27-1215-01

IBM IMS DataPropagator for z/OS

Administrators Guide for Synchronous Propagation

Version 3 Release 1

SC27-1215-01

Note
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 259.

Second Edition (July 2003)

This edition applies to Version 3 Release 1 of IMS DataPropagator for z/OS, 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. Make sure you are using the correct edition for the level of the product.

Abstract

This online book is for people involved in the administration of IMS Synchronous DataPropagator (hereafter referred to as IMS DPROP.) This book covers the following areas:

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

Contents

Abstract iii

Figures ix

About This Book xi

Changes to This Book for IMS DPROP for z/OS Version 3 Release 1 xi
Product Library Changes xi
Types of Data Propagation covered in This Book . . xi
How This Book Is Organized xii
Terms Used in This Book xii
How to Use This Book xiii
What You Should Know xiii

Part 1. IMS DPROP Synchronous Administrative Tasks 1

Chapter 1. Tasks the IMS DPROP Administrator Performs 3

Tasks You Perform for Synchronous Propagation . . 3

Part 2. Mapping and Design of Your IMS DPROP System 7

Chapter 2. Decisions Affecting Mapping and Propagation 11

Propagation Requests and Selecting PRTYPEs . . . 11
Specifying Propagation Direction 11
Selecting a Propagation Request Type 12
PRTYPE=E (Extended Function) 15
PRTYPE=L (Limited Function) 16
PRTYPE=U (User Mapping) 17
PRTYPE=F (Full Function) 18
Mapping Case Characteristics and Rules. 18
Mapping Case 1 19
Mapping Case 2 20
Mapping Case 3 23
User Mapping Cases 30
Mapping Options: Generalized Mapping Cases Only 30
PATH Data 30
WHERE Clause 37

Chapter 3. Propagation Guidelines, Rules, and Restrictions 43

Propagation Guidelines 43
DB2-to-IMS Limitations 43
IMS Logical Relationship Rules. 44
Requirement for a DB2 Primary Key 45
Propagating Variable-Length Segments (IMS-to-DB2) 45
Propagating Variable-Length Segments (DB2-to-IMS) 46
Propagating a Subset of Fields or Columns . . . 47
Mapping Between Fields and Columns 49
Propagating with Multiple Propagation Requests to or from the Same Table 50
Propagating One Segment to or from Multiple Tables 50
Using Propagation Request Sets 50
Defining Propagation Requests with Qualified or Unqualified Table Names. 51
DB2 Referential Integrity Guidelines 53
Defining DB2 RIRs to Match IMS Relationships 54
Using DB2 Delete Rules for Matching RIRs . . . 54
Defining DB2 RIRs for One-Way IMS-to-DB2 Propagation 56
Defining DB2 RIRs for One-Way DB2-to-IMS Propagation 56
Defining DB2 RIRs for Two-Way Propagation . . 57
Implementing Non-matching RIRs for One-Way IMS-to-DB2 and Two-Way Propagation 57
Defining Unique Indexes 57
Unique DB2 Indexes and One-Way IMS-to-DB2 Propagation 58
Truly Unique IMS Secondary Indexes and One-Way DB2-to-IMS Propagation. 58
Unique Indexes and Two-Way Synchronous Propagation 58
Key Mapping Rules by Propagation Request Type 59
Terminology Related to Keys 59
Overview of the Key Mapping Rules 61
Rules For PRTYPE=E (Extended Function) . . . 62
Rules For PRTYPE=L (Limited Function) . . . 69
Comparison of Key Mapping Rules by Propagation Request Type 73
Supported Field Formats and Conversions 74
Describing Fields 75
Converting Data. 76
Summary of Conversion Rules 77
Characteristics of Supported IMS Data Types . . 77
Mapping and Conversion between Numeric Fields 79
Mapping and Conversion between Non-Numeric Data. 82
Normalizing Data 83

Chapter 4. Application Programs Involved in Synchronous Propagation . . 85

IMS/DB2 Mixed-Mode Processing. 85
IMS Application Checkpoint and Restart 85
IMS SETS with ROLS Calls 86
IMS Logical Delete Rules 86
IMS INIT STATUS GROUPA Call 86
ROLB Calls Issued by IMS DPROP 87
BB Status Code (IMS-to-DB2 Propagation) . . . 87
-929 SQL Error Code (DB2-to-IMS Propagation) 87
IMS INIT STATUS GROUPB Call 87

SQL SET CURRENT PACKAGESET Statement. . . 88
Unsupported DB2 Functions in IMS/DB2 Mixed-Mode Environment 88
SQL COMMIT and ROLLBACK Statements . . 88
DB2 Functions Available Only with CAF . . . 88
SQL Statements in PSW Key Other Than 8 or in Authorized State 88

Chapter 5. IMS DPROP Control Information and Environment 89
IMS DPROP Control Information 89
IMS DPROP Directory 89
Propagation Status File 92
IMS DPROP's Use of VLF 92
IMS DPROP's Use of the Global Master Timestamp (GMTS) for Sysplex. 93
How GMTS Works 93
Creating and Updating the GMTS 94
Refreshing or Recreating the VLF PDS 94
JCL Changes for Sysplex IMS DPROP 94
VLF considerations 94
MVG Input Tables 95
Audit Trail Table 95
IMS DPROP Operating Environment 95
Multiple IMS DPROP Systems and Environments 95
Scenarios for One or Multiple IMS DPROP Systems Synchronous 96
IMS Environment 99
Use of DBRC 99
Intersystem Data Sharing 99
DBCTL Support of Changed Data Capture . . 100
Extended Recovery Facility (XRF) Considerations 100
IMS Inserts in Load Mode 100
Database Updates with IMS Utilities 100
DB2 Environment 101
SQL Updates in a Non-IMS Environment . . . 101
Remote SQL Updates to Propagated Tables . . 101
Table Updates with DB2 Utilities 101
CICS Environment 101
Coordinating Availability of IMS Databases and DB2 Tables 102
Reducing Operational Risks Using ERROPT=IGNORE 102

Part 3. Setting Up for Data Propagation 103

Chapter 6. Setting Up Your Systems for Synchronous Propagation 105
Creating or Changing DBDs 105
EXIT Keyword (IMS-to-DB2) 106
Specifying the VERSION Keyword 111
Defining the PCBs Reserved for HUP (DB2-to-IMS Synchronous Propagation) 112
Increasing CPU Time Limits of Transactions . . . 113
Converting DB2-Only Programs to Mixed-Mode IMS/DB2 Programs (DB2-to-IMS) 114
Preparing DB2 for Data Propagation for DB2-to-IMS Propagation 114
Binding DB2 Plans: Initial Bind 115
Binding Plans with DB2 Package Bind 115
Binding Plans without DB2 Package Bind . . . 116
Creating DB2 Tables 116
Specifying Columns 116
Table Qualification. 117
Protecting Propagated Tables from Nonpropagating SQL Updates 117
One-Way IMS-to-DB2 Propagation 117
DB2-to-IMS and Two-Way Synchronous Propagation 117
Identifying to DB2 the Tables Subject to Data Capture (DB2-to-IMS Synchronous Propagation). . 118
Binding DB2 Plans for IMS-to-DB2 Synchronous Propagation: Subsequent Bind 118
Starting DB2 Monitor Trace Class 6 for DB2-to-IMS Propagation 119

Chapter 7. Defining and Changing Propagation Requests 121
Defining Propagation Requests Using DataRefresher 121
CREATE DATATYPE Command 122
CREATE DXTPSB Command 122
CREATE DXTVIEW Command 123
SUBMIT Command and EXTRACT Statement 124
DataRefresher and User Mapping Cases . . . 127
Defining Propagation Requests Using the MVG Input Tables 128
Identifying the Propagation Request 128
Specifying the IMS Segments to be Propagated 129
Specifying the DB2 Tables 129
Specifying the Fields 129
Executing the MVGU. 129
Propagation Parameters 131
PRTYPE—Type of Propagation Request. . . . 132
MAPCASE—Mapping Case 132
PATH—Path Data Option 132
MAPDIR—Mapping Direction. 133
TABQUAL2—DB2 Table Qualifier Used for Validation 133
ERROPT—Error Option 133
MAXERROR—Maximum Number of Reported Propagation Errors 133
ACTION 133
PRSET—Propagation Request Set Name . . . 134
PROPSUP—Propagation Suppression 134
AVU—Avoid Unnecessary Updates 134
DEFVEXT—Default Value Extension Segments: Mapping Case 2 DB2-to-IMS Only 135
KEYORDER—DB2 Key Ordering Sequence . . 135
PERFORM—Type of Operation: DataRefresher only 135
EXITNAME—Name of Propagation Exit . . . 135
PROPSEGM—Propagated Segments: User Mapping with DataRefresher Only 135
PCBLABEL—Label of IMS PCB for DB2-to-IMS Propagation Only 136
BIND—Options for a DB2 Package Bind . . . 136

Deleting a Propagation Request 136
Replacing a Propagation Request 137
Rebuilding a Propagation Request 137
Revalidating Propagation Requests 137

Chapter 8. Granting Privileges and Authorizations for DB2 Objects . . . 139
IMS DPROP Tables, Utilities, and Related Objects 139
Granting Privileges for IMS DPROP Tables . . 140
Binding Packages of IMS DPROP Modules . . 141
Granting Privileges for IMS DPROP Collections 141
Binding Plans of IMS DPROP Utilities 142
Running IMS DPROP Utilities 142
Propagated Tables, Propagating Applications, and Related Objects 144
Granting Table Privileges for Propagated Tables 144
Granting Privileges for Propagating Collections 146
Binding Packages of SQL Update Modules and Propagation Exit Routines 146
Binding SQL Update Modules into Different Packages 147
Binding DB2 Plans of Propagating Applications 147
Running Propagating Applications 148

Chapter 9. Binding and Administering Plans 149
Binding Plans with Bind Package 149
Using Different Collection IDs 150
Job Stream for Binding DB2 Packages 150
Job Stream for Binding DB2 Plans with Bind Package 152
Binding Plans without Bind Package 153
Binding Synchronous Propagation Applications 153
Binding the User Asynchronous Receiver Program 154
Job Stream for Binding DB2 Plans without Bind Package 154
DB2 ALIAS and SYNONYM Statements . . . 156
Administering DB2 Plans with or without a Resource Translation Table (RTT) 158

Chapter 10. Extracting and Loading Data for IMS-to-DB2 Propagation . . . 159
Overview of the Extract and Load Process . . . 159
Preventing Updates to IMS Databases 159
Using Status Change Utility (SCU) 160
Alternative to Using SCU 160
Doing the Extract and Load with DataRefresher 161
Doing the Extract and Load with Your Programs 163

Chapter 11. Extracting and Loading Data for DB2-to-IMS (DLU) Propagation 165
Overview 165
DLU Restrictions 166
DLU Input and Output 166
DLU Input 166
DLU Output 167
How the DLU Selects and Processes Input Data 167
Simple Scenario 169
Complex Scenarios 169
Considerations for Segments without a Unique DL/I Key 171
Considerations for Paired Segment Types 171
Physically Paired Segment Types 171
Virtually Paired Segment Types 172

Part 4. Propagating Data with IMS DPROP 173

Chapter 12. Performing Synchronous Propagation 175
Normal RUP Processing 175
Environment 175
Processing 175
Normal HUP Processing 176
HUP Environment 176
HUP Processing 176
Error Handling Options 177
Dynamic Backout in IMS Environments . . . 178
DB2 Region Error Option 178
IMS DPROP Error Option 178
IMS INIT STATUS Call 179
RUP and HUP Error Processing 183
Severe Errors 185
DB2 Deadlocks 185
IMS Deadlocks 185
Propagation Emergency Stopped or Deactivated 185
Unavailable Resources 186
Other Errors 186
Summary of Error Handling 186
Some Causes of Unavailable Resources 187
RUP and HUP Error Reporting 188
Limiting the Number of Error Messages Resulting From ERROPT=IGNORE 188
Using MVS to Suppress Messages 189

Chapter 13. Controlling Synchronous Propagation States 191
Synchronous Propagation States and Modes . . . 191
Synchronous Propagation State of the Entire IMS DPROP System 191
Synchronous Propagation Status of Individual Propagation Requests 191
PROP OFF Mode for DB Repair Programs . . . 192
Read-Only Status of IMS Databases 193
Read-Only Access Mode of DB2 Table Spaces and Databases 194
Status Change Utility (SCU) 194
Controlling Propagation Requests 195
Controlling Full-Function IMS Databases . . . 200
Controlling DB2 Databases and Table Spaces 201
Controlling the IMS DPROP System 201
General Service Functions of the SCU 202
RUP and HUP Control Statements 205
Controlling Synchronous Propagation Using PROP Control Statements 206
Controlling Traces 207

Controlling the Number of Resident SQL Update Modules and PRCBs 208

Chapter 14. Database Maintenance for Synchronous Propagation 211
Checkpoint and Restart in the IMS and DB2 Environment 211
Restart of IMS Online and DB2 212
Checkpoint and Restart of an IMS Batch Program 212
Database Backout for IMS Batch Programs . . . 212
IMS Dynamic Backout for Batch Regions . . . 212
Backout of Committed Data 212
Backup and Recovery 213
System Data Sets 213
Databases 213
Timestamp Recovery 214
Data Resynchronization 214
Database Repair 215
IMS and DB2 Repair Functions 215
User-Written Repair Programs 215
Preventing Inadvertent Execution of Repair Programs 216
Database Reorganization and Load 216
Initial Load of IMS Databases 217
Load of DB2 Tables 217
CCU Verification 217
IMS DPROP Directory Recovery 217

Chapter 15. Verifying Data Consistency (CCU) 219
Overview of the CCU 219
When to Use the CCU 220
CCU Considerations for Synchronous Propagation 221
Considerations When Concurrent Updates Are Being Done 221
Data Availability 221
DB2 Referential Integrity Constraints 221
Running the CCU 222
Phases of the CCU 222
CCU Verification Techniques 222
Types of Inconsistencies and Generated Repair Statements 223
Large Numbers of Inconsistencies 224
Some Reasons for Inconsistencies 224

Chapter 16. IMS DPROP's Problem Determination Tools 227
IMS DPROP Trace Facilities 227
IMS DPROP Audit Facilities 228
Using SMF 228
Audit Extract Utility and Audit Trail Table . . 228
Creating an Audit Trail 229
Audit Trail Table Security 230
Comparison of Audit and Trace Information . . 230
CCU and the Audit Trail 230
Monitoring Consistency with the CCU 231
Monitoring Propagation with the Message Table of the IMS DPROP Directory 231

Chapter 17. IMS DPROP Performance and Monitoring. 233
IMS DPROP Performance 233
Mapping and Design Phase 233
Setup Phase 234
Propagation Phase: Synchronous Propagation Performance 234
Propagation Phase: User Asynchronous Propagation Performance 237
CCU Execution 237
Monitoring Propagation 238

Part 5. Appendixes 241

Appendix A. JCL Information 243
JCL Changes for Synchronous Propagation . . . 243
JCL Changes for DB2 245
DB2 JCL Changes in the IMS Control Region 245
DB2 JCL Changes in IMS Dependent Regions 245
DB2 JCL Changes in IMS Batch Regions . . . 246
SSM Member in PROCLIB 248

Appendix B. Language Interface and Multiple DB2 Systems. 251

Appendix C. Synchronous Propagation Storage Requirements . . 253
Virtual Storage Requirements 253
Installation-Independent Requirements 253
Installation-Sensitive Requirements 253
Transient Storage Requirements 254
Real Storage Requirements 255

Appendix D. Converting PRTYPE=F into PRTYPE=E Propagation Requests 257

Notices 259
Programming Interface Information 261
Trademarks 261

Glossary of Terms and Abbreviations 263

Bibliography. 273
The IMS DataPropagator for z/OS Version 3 Release 1 Library 273
Other Books Referenced in This Book 273

Index 275

Figures

1. Mapping Case 1 . . . 19
2. Mapping Case 2 . . . 21
3. Mapping Case 3 . . . 25
4. Containing Segment and Internal Segment Type . . . 26
5. Conceptually Normalizing the Database for Mapping Case 3 . . . 27
6. Mapping Case 1 Propagation Request Propagating PATH Data . . . 31
7. Denormalization of Data with PATH Data . . . 33
8. Identifying Parent/Ancestors Contributing Modifiable PATH Data to PR3 . . . 35
9. PR Propagating ID Fields of a Physical Parent/Ancestor as PATH Data . . . 37
10. Mapping with a WHERE Clause . . . 38
11. Defining Variable-Length Segments . . . 46
12. Mapping Unique IMS Fully Concatenated Keys to DB2 Primary Keys with PRTYPE=Es (Ideal Case) . . . 66
13. Mapping Unique Conceptual Fully Concatenated Keys to Primary DB2 Keys with PRTYPE=E (Non-Ideal Case) . . . 68
14. Mapping of Keys with PRTYPE=L . . . 72
15. IMS DPROP Directory . . . 91
16. EKYGMTS DD statement . . . 94
17. IMS DPROP Scenario 1 . . . 96
18. IMS DPROP Scenario 2 . . . 97
19. IMS DPROP Scenario 3 . . . 98
20. IMS DPROP Scenario 4 . . . 98
21. IMS DPROP Scenario 5 . . . 99
22. Adding HUP PCBs to an Existing PSB . . . 113
23. Propagation Request Definition with DataRefresher . . . 127
24. Propagation Request Definition with MVG Input Tables . . . 131
25. Columns That Can Be Updated in Propagated DB2 Tables . . . 145
26. BIND PACKAGE Job Stream for IMS DPROP . . . 151
27. BIND PLAN Job Stream When Using Packages . . . 152
28. BIND Plan Job Stream without Packages . . . 155
29. Two-Step BIND Process . . . 157
30. Using the DB2 CREATE ALIAS Statement . . . 157
31. Using the DB2 CREATE SYNONYM Statement . . . 158
32. Extract and Load Process Using DataRefresher . . . 162
33. Extract and Load Process with User-Written Programs . . . 164
34. Overview of DLU Processing . . . 168
35. Error Processing Logic of RUP and HUP . . . 184
36. CCU Execution and the Repair Process . . . 220
37. Overview of the IMS DPROP Audit Process . . . 229
38. Relinking IMS DPROP Module EKYY371X . . . 251

About This Book

This book is for people who administer IMS™ Synchronous DataPropagator™ z/OS® Version 3 Release 1. Administration responsibilities include the design, implementation, and control of data propagation, as well as operation in the data propagation environment. This book describes the tasks you, as administrator, perform for IMS DPROP Synchronous Propagation. This softcopy book is available only in PDF and BookManager® formats. This book is available on the z/OS Software Products Collection Kit, SK3T-4270. You can also get the most current versions of the PDF and BookManager formats by going to the IBM® Data Management Tools Web site at www.ibm.com/software/data/db2imstools and linking to the Library page.

Changes to This Book for IMS DPROP for z/OS Version 3 Release 1

This edition, which is available in softcopy only, includes technical and editorial changes. The terminology for DPROP Asynchronous has been changed to LOG-ASYNC while functionally remaining the same. Major functional changes are with the addition of MQSeries® Asynchronous Propagation, and a separate MQSeries Administrators Guide.

Product Library Changes

The IMS DPROP Version 3.1 library has been updated with information about MQSeries Asynchronous Propagation. There are now three Administrator Guides, one for each primary mode of propagation:

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

A new book, *IMS DataPropagator for z/OS: Concepts* has been added to the library which replaces part one of the previous Administration Guide (2.2) and provides a conceptual description of all the modes of data propagation.

Types of Data Propagation covered in This Book

This book covers only Synchronous propagation. For other types of propagation, see the appropriate Administrators Guide.

Synchronous propagation

To propagate changes in the following directions:

- One-way IMS to DB2®

  Changes made to a set of IMS databases are propagated to a corresponding set of DB2 databases. IMS changes are applied to the DB2 tables within the same unit of work (UOW).
- One-way DB2 to IMS

  Changes made to a set of DB2 databases are propagated to a corresponding set of IMS databases. DB2 changes are applied to the IMS databases within the same UOW.
- Two-way (IMS to DB2 and DB2 to IMS)

Changes made to either set of databases (IMS or DB2) are propagated to the corresponding set of databases (DB2 or IMS, respectively). Changes are applied within the same UOW.

How This Book Is Organized

This book is divided into five parts:

- Part 1, "IMS DPROP Synchronous Administrative Tasks," on page 1, presents a summary list of the administrator tasks. Part 1 consists of chapter 1.
- Part 2, "Mapping and Design of Your IMS DPROP System," on page 7, covers the mapping and definition phase of data propagation. It describes the decisions you must make including rules and guidelines to follow, as you design your IMS DPROP environment. Part 2 consists of chapters 2-5.
- Part 3, "Setting Up for Data Propagation," on page 103, 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 6-12.
- Part 4, "Propagating Data with IMS DPROP," on page 173, covers the actual propagation phase including the maintenance and control phase of data propagation. It describes tasks to operate, maintain, and tune IMS DPROP. Part 4 consists of chapters 13-21.
- Part 5 offers additional information about JCL, storage requirements, and other aspects of data propagation. It consists of four appendixes A-D.
  - Appendix A, "JCL Information," on page 243, describes the JCL changes you need to make in the IMS DPROP environment.
  - Appendix B, "Language Interface and Multiple DB2 Systems," on page 251, describes the DB2 language interfaces and use of multiple DB2 systems.
  - Appendix C, "Synchronous Propagation Storage Requirements," on page 253, describes IMS DPROP's storage requirements in IMS regions doing synchronous propagation.
  - Appendix D, "Converting PRTYPE=F into PRTYPE=E Propagation Requests," on page 257, tells IMS DPROP R1 users how to convert IMS DPROP R1 TYPE=F to the more powerful IMS DPROP R2 TYPE=E.

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

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

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

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

How to Use This Book

Key administrative tasks for design, setup and implementation of data propagation are listed in Chapter 1, “Tasks the IMS DPROP Administrator Performs,” on page 3. The order in which tasks are presented is the recommended order but not required.

What You Should Know

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

This book also assumes you have read and understand the *IMS DataPropagator for z/OS: Concepts* book which provides a conceptual description of data propagation and the various modes.

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

Part 1. IMS DPROP Synchronous Administrative Tasks

Chapter 1. Tasks the IMS DPROP Administrator Performs 3
Tasks You Perform for Synchronous Propagation . . 3

Part 1 identifies the tasks associated with administering Synchronous IMS DPROP. Chapter 1, "Tasks the IMS DPROP Administrator Performs," on page 3, lists the tasks you, as IMS DPROP administrator, perform to: design, setup, and monitor propagation.

Chapter 1. Tasks the IMS DPROP Administrator Performs

This chapter summarizes the tasks you perform to:
- Map data and design for propagation
- Set up IMS DPROP, IMS, and DB2
- Propagate data
- Maintain and control your propagation environment

Each task that is listed in the table is followed by a reference to 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 it is not not required that you follow it. You might also repeat tasks in various phases of propagation. For example, you extract and load data during setup phase but you also can extract and load data during the maintenance and control phase in order to synchronize your data.

Tasks You Perform for Synchronous Propagation

Table 1 summarizes tasks you complete in order to prepare for and implement synchronous propagation. The tasks are divided into these phases:

- Mapping and design—determining the data that you are propagating
- Setup—ensuring that your IMS and DB2 systems are ready for propagation
- Propagation—beginning and refining propagation
- Maintenance and control—periodically checking for data consistency or adjusting propagation requests

Table 1. Summary of Task Steps for Synchronous Propagation

	Synchronous Propagation		
Administrator Tasks	**IMS to DB2**	**DB2 to IMS**	**Two Way**
1. Install IMS DPROP. Refer to *IMS DataPropagator Installation* for details.	Y	Y	Y
Mapping and Design Phase			
2. In IMS DPROP, define the mappings. • Identify mapping requirements. • Select the IMS segments and data fields that are to be propagated. • Identify DB2 tables and columns. • Determine the mapping case or design your own mapping. See Part 2, "Mapping and Design of Your IMS DPROP System," on page 7.	Y	Y	Y

Table 1. Summary of Task Steps for Synchronous Propagation (continued)

Administrator Tasks	Synchronous Propagation		
	IMS to DB2	**DB2 to IMS**	**Two Way**
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 99. Also see "Supported Field Formats and Conversions" on page 74 for specifics on field specifications and "Propagating Variable-Length Segments (IMS-to-DB2)" on page 45 and "Propagating Variable-Length Segments (DB2-to-IMS)" on page 46 for specific rules and limitations.	Y	-	Y
4. Create exit routines, if needed. Exits must be coded and compiled into the IMS DPROP library. Refer to the *IMS DataPropagator Customization Guide* for details on creating exit routines.	Y	Y	Y
Setup Phase			
5. In IMS, register each database that is a source or target of propagation data with DBRC using the INIT.DB command. This step is recommended for synchronous propagation See "Use of DBRC" on page 99 for an explanation of this requirement. Refer to *IMS Utilities Reference: System* for details on the INIT.DB command.	Y	Y	Y
6. In IMS, (create or) modify the database definition (DBD) of each database to call the RUP. See "Creating or Changing DBDs" on page 105. You are identifying the segments that are subject to propagation; 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.	Y	-	Y
7. In IMS, modify JCL of IMS control regions, dependent regions and batch regions. For more information, see: • "JCL Changes for Synchronous Propagation" on page 243 • "DB2 JCL Changes in IMS Dependent Regions" on page 245 • "DB2 JCL Changes in IMS Batch Regions" on page 246	Y	Y	Y
8. For DB2-to-IMS propagation, in IMS DPROP define in the PSBs the PCBs reserved for HUP. See "Defining the PCBs Reserved for HUP (DB2-to-IMS Synchronous Propagation)" on page 112.	-	Y	Y
9. In IMS, increase the CPU time limits of transactions and applications. See "Increasing CPU Time Limits of Transactions" on page 113 for an explanation and instructions.	Y	Y	Y
10. In DB2, if required, convert DB2-only programs to mixed-mode IMS/DB2 programs. See "IMS/DB2 Mixed-Mode Processing" on page 85 for a discussion and "Converting DB2-Only Programs to Mixed-Mode IMS/DB2 Programs (DB2-to-IMS)" on page 114 for procedures.	-	Y	Y

Table 1. Summary of Task Steps for Synchronous Propagation (continued)

Administrator Tasks	Synchronous Propagation		
	IMS to DB2	**DB2 to IMS**	**Two Way**
11. For DB2-to-IMS synchronous propagation, prepare DB2 for propagation. See "Preparing DB2 for Data Propagation for DB2-to-IMS Propagation" on page 114.	-	Y	Y
12. In DB2, do an initial DB2 bind of the application plans. See "Binding DB2 Plans of Propagating Applications" on page 147 for authorization issues and "Binding DB2 Plans: Initial Bind" on page 115 for specifics about binding for synchronous propagation. (You are binding for applications that change data that is eventually propagated.) Also see Chapter 9, "Binding and Administering Plans," on page 149, for information on how to bind.	Y	Y	Y
13. If the propagated DB2 tables do not already exist, create them in DB2. The tables must exist before you use MVGU to create propagation requests. See "Creating DB2 Tables" on page 116 additional information.	Y	Y	Y
14. In DB2, protect propagated DB2 tables from inadvertent SQL updates. See "Protecting Propagated Tables from Nonpropagating SQL Updates" on page 117.	Y	Y	Y
15. For DB2-to-IMS synchronous propagation, in DB2 identify to DB2 the tables subject to data capture See "Identifying to DB2 the Tables Subject to Data Capture (DB2-to-IMS Synchronous Propagation)" on page 118.	-	Y	Y
16. Use the DataRefresher MCE or the IMS DPROP MVGU to: • Populate MVG tables with propagation request definitions. • Build propagation requests. The initial state of propagation requests is INACTIVE. See Chapter 7, "Defining and Changing Propagation Requests," on page 121.	Y	Y	Y
17. 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 application plans. See "Binding DB2 Plans of Propagating Applications" on page 147 for authorization issues, "Binding DB2 Plans for IMS-to-DB2 Synchronous Propagation: Subsequent Bind" on page 118, and Chapter 9, "Binding and Administering Plans," on page 149, for information on how to bind.	Y	-	Y
18. Extract and load data. Use SCU and DataRefresher. See Chapter 10, "Extracting and Loading Data for IMS-to-DB2 Propagation," on page 159, and Chapter 11, "Extracting and Loading Data for DB2-to-IMS (DLU) Propagation," on page 165.	Y	Y	Y
19. Use the SCU to activate propagation requests. Refer to the *IMS DataPropagator Reference* for information on the SCU and the ACTIVATE statement.	Y	Y	Y
20. For DB2-to-IMS synchronous propagation, start DB2 monitor trace class 6. See "Starting DB2 Monitor Trace Class 6 for DB2-to-IMS Propagation" on page 119.	-	Y	Y
Propagation Phase (Regularly Scheduled Activity)			

Table 1. Summary of Task Steps for Synchronous Propagation (continued)

Administrator Tasks	Synchronous Propagation		
	IMS to DB2	**DB2 to IMS**	**Two Way**
No activity required.	Y	Y	Y
Maintenance and Control Phase (Periodic Activity)			
21. Optional: Use the CCU to compare IMS and DB2 data and check for consistency. See Chapter 15, "Verifying Data Consistency (CCU)," on page 219, to understand how you can use CCU. Also refer to the *IMS DataPropagator Reference* for detailed information about the CCU.	Y	Y	Y
22. Delete, replace, and rebuild propagation requests, as needed. See information on deleting, replacing, rebuilding, and revalidating propagation requests beginning on page 136.	Y	Y	Y
23. Revalidate propagation requests, especially if changes have been made to either the IMS DBDs or DB2 tables. See "Revalidating Propagation Requests" on page 137. Use MVGU to run a REVALIDATE function. The MVGU is described in detail in the *IMS DataPropagator Reference*.	Y	Y	Y

Part 2. Mapping and Design of Your IMS DPROP System

Chapter 2. Decisions Affecting Mapping and Propagation 11
Propagation Requests and Selecting PRTYPEs . . . 11
Specifying Propagation Direction 11
Selecting a Propagation Request Type 12
PRTYPE=E (Extended Function) 15
PRTYPE=L (Limited Function) 16
PRTYPE=U (User Mapping) 17
PRTYPE=F (Full Function) 18
Mapping Case Characteristics and Rules. 18
Mapping Case 1 19
Mapping Case 2 20
Rules for Mapping Fields in Extension Segments 21
Extension Segment for DB2-to-IMS Propagation and PRTYPE=E. 22
DB2-to-IMS Propagation to Extension Segments 22
Mapping Case 3 23
Definitions: Containing and Internal Segments 26
Mapping Design for Mapping Case 3. . . . 26
Internal Segments and Segment Exit Routines 28
Unique Identification of Internal Segments . . 28
Fixed/Variable Number of Occurrences of Internal Segments 28
PRTYPE=E and Internal Segments 29
DB2-to-IMS Propagation of Internal Segments 29
DLU Processing of Internal Segments. . . . 30
SEG= Keyword on IMS DPROP Control Statements. 30
User Mapping Cases 30
Mapping Options: Generalized Mapping Cases Only 30
PATH Data 30
Uses of PATH Data 31
PATH=DENORM: Denormalizing Data to Improve Performance of DB2 Queries . . . 32
PATH=ID: Mapping ID Fields of a Physical Parent/Ancestor. 35
WHERE Clause 37
Selective Propagation Using the WHERE Clause 38
Fields That Can Be Included in the WHERE Clause 39
Fields That Cannot Be Included in the WHERE Clause 39
Conditions and Operators Used with the WHERE Clause 40
Recommendations for Propagating Parent Segments with a WHERE Clause 40
Recommendation for Propagating Logical Parent Segments with a WHERE Clause . . . 41

Chapter 3. Propagation Guidelines, Rules, and Restrictions 43
Propagation Guidelines 43
DB2-to-IMS Limitations 43
IMS Logical Relationship Rules. 44
Paired Logical Children 44
Delete Rules 44
Requirement for a DB2 Primary Key 45
Propagating Variable-Length Segments (IMS-to-DB2) 45
Propagating Variable-Length Segments (DB2-to-IMS) 46
Propagating a Subset of Fields or Columns . . . 47
Propagation of a Subset of Fields in a Segment 47
Propagating a Subset of Columns in a Table 48
Mapping Between Fields and Columns 49
Mapping One Field to Multiple Columns . . 49
Mapping Multiple Fields to One Column . . 49
Propagating with Multiple Propagation Requests to or from the Same Table 50
Propagating One Segment to or from Multiple Tables 50
PRTYPE=L and One-Way IMS-to-DB2 Propagation 50
PRTYPE=E and DB2-to-IMS Propagation . . 50
PRTYPE=U 50
Using Propagation Request Sets 50
Examples of Propagation Request Set Use . . 51
Defining Propagation Requests with Qualified or Unqualified Table Names. 51
Qualified Table Names 51
Unqualified Table Names. 52
DB2 Referential Integrity Guidelines 53
Defining DB2 RIRs to Match IMS Relationships 54
Using DB2 Delete Rules for Matching RIRs. . . 54
RIR Matching a Physical IMS Parent/Child Relationship 55
RIR Matching a Logical IMS Parent/Child Relationship 55
Defining DB2 RIRs for One-Way IMS-to-DB2 Propagation 56
Defining DB2 RIRs for One-Way DB2-to-IMS Propagation 56
Defining DB2 RIRs for Two-Way Propagation . . 57
Implementing Non-matching RIRs for One-Way IMS-to-DB2 and Two-Way Propagation 57
Defining Unique Indexes 57
Unique DB2 Indexes and One-Way IMS-to-DB2 Propagation 58
Truly Unique IMS Secondary Indexes and One-Way DB2-to-IMS Propagation. 58
Unique Indexes and Two-Way Synchronous Propagation 58
Key Mapping Rules by Propagation Request Type 59
Terminology Related to Keys 59
Overview of the Key Mapping Rules 61
Rules For PRTYPE=E (Extended Function) . . . 62
Example of Mapping Keys in Ideal Case (PRTYPE=E) 65

Example of Mapping Keys in Non-Ideal Case (PRTYPE=E) 67
Rules For PRTYPE=L (Limited Function) . . . 69
Example of Mapping Keys (PRTYPE=L) . . . 71
Comparison of Key Mapping Rules by Propagation Request Type 73
Supported Field Formats and Conversions 74
Describing Fields 75
Converting Data 76
Summary of Conversion Rules 77
Characteristics of Supported IMS Data Types . . 77
Mapping and Conversion between Numeric Fields 79
Mapping and Conversion between Binary Integers. 80
Mapping and Conversion between Decimal Fields 80
Mapping and Conversion between Binary Integers and Decimal Fields 81
Mapping and Conversion between Floating Point Numbers 82
Mapping and Conversion between Non-Numeric Data 82
Mapping and Conversion between Character/Graphic Strings 82
Mapping and Conversion between Dates . . 83
Mapping and Conversion between Times . . 83
Mapping and Conversion between Timestamps 83
Normalizing Data 83

Chapter 4. Application Programs Involved in Synchronous Propagation 85
IMS/DB2 Mixed-Mode Processing 85
IMS Application Checkpoint and Restart 85
IMS SETS with ROLS Calls 86
IMS Logical Delete Rules 86
IMS INIT STATUS GROUPA Call 86
ROLB Calls Issued by IMS DPROP 87
BB Status Code (IMS-to-DB2 Propagation) . . . 87
-929 SQL Error Code (DB2-to-IMS Propagation) 87
IMS INIT STATUS GROUPB Call 87
SQL SET CURRENT PACKAGESET Statement. . . 88
Unsupported DB2 Functions in IMS/DB2 Mixed-Mode Environment 88
SQL COMMIT and ROLLBACK Statements . . 88
DB2 Functions Available Only with CAF . . . 88
SQL Statements in PSW Key Other Than 8 or in Authorized State 88

Chapter 5. IMS DPROP Control Information and Environment 89
IMS DPROP Control Information 89
IMS DPROP Directory 89
Propagation Status File 92
IMS DPROP's Use of VLF 92
VLF Requirements 92
Initializing, Refreshing or Recreating VLF Objects 93
IMS DPROP's Use of the Global Master Timestamp (GMTS) for Sysplex. 93
How GMTS Works 93
Creating and Updating the GMTS 94
Refreshing or Recreating the VLF PDS 94
JCL Changes for Sysplex IMS DPROP 94
VLF considerations 94
MVG Input Tables 95
Audit Trail Table 95
IMS DPROP Operating Environment 95
Multiple IMS DPROP Systems and Environments 95
Scenarios for One or Multiple IMS DPROP Systems Synchronous 96
Scenario 1 96
Scenario 2 97
Scenario 3 97
Scenario 4 98
Scenario 5 98
IMS Environment 99
Use of DBRC 99
Intersystem Data Sharing 99
DBCTL Support of Changed Data Capture . . 100
Extended Recovery Facility (XRF) Considerations 100
IMS Inserts in Load Mode 100
Database Updates with IMS Utilities 100
DB2 Environment 101
SQL Updates in a Non-IMS Environment . . . 101
Remote SQL Updates to Propagated Tables . . 101
Table Updates with DB2 Utilities 101
CICS Environment 101
Coordinating Availability of IMS Databases and DB2 Tables 102
Reducing Operational Risks Using ERROPT=IGNORE 102

Part 2 covers the mapping and definition phase of data propagation. It consists of four chapters:

- Chapter 2, "Decisions Affecting Mapping and Propagation," on page 11, provides information to help you make decisions about 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
- Chapter 3, "Propagation Guidelines, Rules, and Restrictions," on page 43, presents guidelines and rules for mapping data and advises you on restrictions that affect propagation. Topics in this chapter include:

  - relationship rules
  - propagation of subsets
  - key mapping rules
  - supported field formats and conversions
- Chapter 4, "Application Programs Involved in Synchronous Propagation," on page 85, discusses considerations that can help you efficiently operate your mixed-mode application programs with IMS DPROP. Topics include:
  - checkpoint and restart
  - logical delete rules
  - the applicability of SETS, ROLS, and INIT STATUS calls
- Chapter 5, "IMS DPROP Control Information and Environment," on page 89, describes IMS DPROP control information, such as IMS DPROP tables and files, and provides operational environment scenarios that help you understand and prepare for an IMS DPROP environment.

Chapter 2. Decisions Affecting Mapping and Propagation

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

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

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

See Chapter 7, "Defining and Changing Propagation Requests," on page 121 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. You can specify:

HR Hierarchical-to-relational propagation, which is one-way IMS-to-DB2 propagation.

RH Relational-to-hierarchical synchronous propagation, which is one-way DB2-to-IMS propagation.

TW [1] Two-way propagation, which is IMS-to-DB2 and DB2-to-IMS propagation.

The propagation direction supported by IMS DPROP depends on the propagation request type.

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[2]

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 LOG-ASYNC or user asynchronous propagation with the IMS Asynchronous Data Capture function. Consequently, your SQL updates to the tables propagated by synchronous TW propagation requests are propagated to IMS, but are not propagated to the other tables propagated by the HR propagation requests.

  Therefore, when you do SQL updates to the tables propagated by TW propagation requests, you must apply the equivalent SQL updates to the tables propagated by HR propagation requests to ensure consistency between:
  - The tables propagated by the HR propagation requests
  - The IMS segment and 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 the following types of Synchronous propagation: • IMS-to-DB2 • DB2-to-IMS • Two-way
L	Limited function propagation requests, used with generalized mapping. It supports only IMS-to-DB2 propagation.
U	User propagation request, used with user mapping and Propagation exit routines.
F	Full function propagation request, used with previous IMS DPROP releases.

1. IMS DPROP considers it an error if IMS Data Capture invokes RUP for a segment propagated by RH propagation requests. Do not use DL/I calls to insert or delete IMS segments propagated by active RH propagation requests; and do not use DL/I REPL calls to change the value of fields propagated by active RH propagation requests (but you can use REPL calls to change the value of nonpropagated fields).
2. 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.

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 propagation.

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

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

However, PRTYPE=L only supports IMS-to-DB2 synchronous propagation (does not support DB2-to-IMS synchronous propagation or 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 will 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)

Table 2. Characteristics of Propagation Request Types (continued)

	PRTYPE=E	**PRTYPE=L**	**PRTYPE=U**
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
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)	Yes (see PRTYPE=L note 1)	N/A
Segment and Field exit routines must support mapping for DB2-to-IMS 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 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 13.

PRTYPE=E has the following characteristics:

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

   The WHERE clause option is supported.

   The PATH data option is supported. PATH data must not change its value and you must define your propagation request with PATH=ID. Refer to "PATH Data" on page 30 for more details on this subject.
2. IMS-to-DB2 propagation, DB2-to-IMS propagation, and two-way propagation are all supported.
   - For DB2-to-IMS and two-way propagation:
     a. 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 propagation.
     b. 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.
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 59.
7. IMS DPROP checks that DB2 RIRs are compatible with IMS parent/child relationships. See "DB2 Referential Integrity Guidelines" on page 53 for further discussion.

   For one-way IMS-to-DB2 propagation, DB2 RIRs are optional. For one-way DB2-to-IMS and two-way synchronous propagation, you should implement matching DB2 RIRs.
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 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 13.

PRTYPE=L has the following characteristics:

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

   The WHERE clause option is supported.

   The PATH data option is supported. PATH data is supported both for the PATH=ID option (which requires that PATH data not change its value) and for the PATH=DENORM option (which allows PATH data to change its value). See “PATH Data” on page 30 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 59.
7. IMS DPROP checks that DB2 RIRs are compatible with IMS parent/child relationships. See "RIR Matching a Physical IMS Parent/Child Relationship" on page 55 for a complete discussion.

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

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

PRTYPE=U (User Mapping)

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

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

1. A Propagation exit, that you write, maps, converts, and propagates data.
2. Your Propagation exit routine provides support for IMS-to-DB2 propagation, DB2-to-IMS synchronous propagation, and two-way synchronous propagation.
   - For DB2-to-IMS and two-way 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.
     - 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
- 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 (DPROPNR) 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 3.1, 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, (for example, DB2-to-IMS propagation), convert your PRTYPE=F to PRTYPE=E because the R2 rules for PRTYPE=E are stricter than the R1 rules for PRTYPE=F. You might also need to modify your IMS database and DB2 table definitions. See Appendix D, “Converting PRTYPE=F into PRTYPE=E Propagation Requests,” on page 257.
- 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 *Customization Guide* for more information on exit routines.

Mapping Case 1

Mapping case 1 propagates one single segment type occurrence to or from a row in a single DB2 table. The segment type being propagated is called the *entity segment*, and 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 1, 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 1 would simply be reversed.

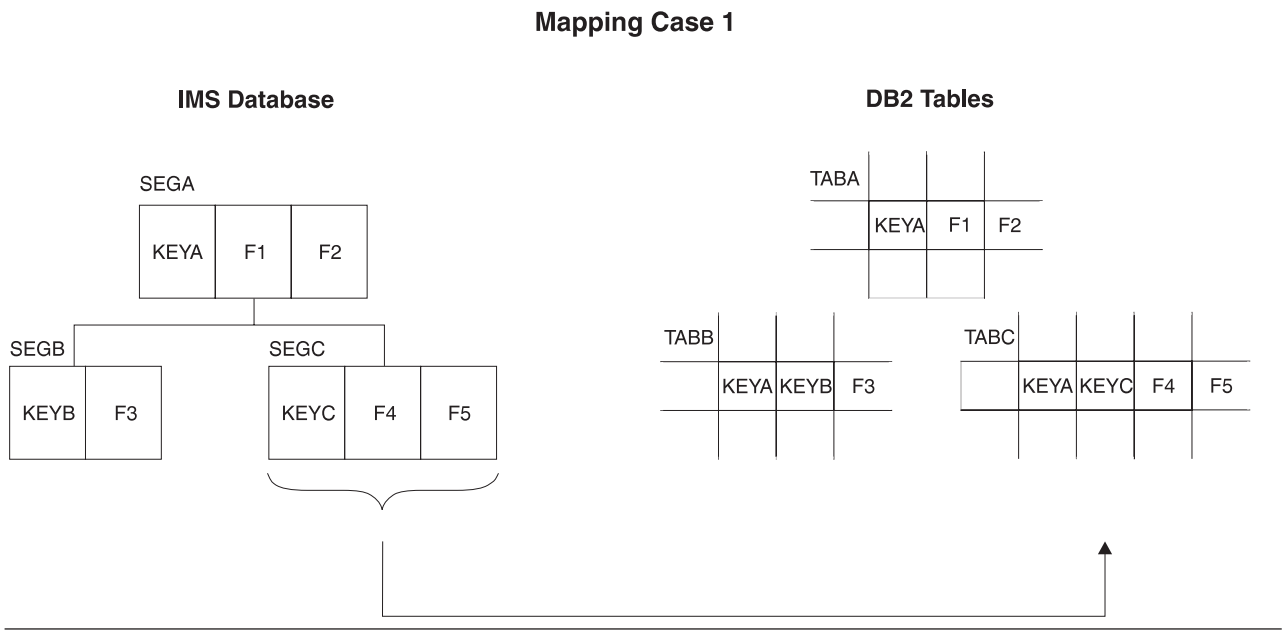


Figure 1. Mapping Case 1

Optional:

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

See “Mapping Options: Generalized Mapping Cases Only” on page 30 for a detailed description of these options.

Mapping Case 2

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

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

In Figure 2 on page 21, mapping case 2 maps one single segment type occurrence with the keys of the parent and all ancestors up to the root. the mapping case also maps data from one or more immediately subordinate segment types (with a maximum of one occurrence of each segment type per parent).

In Figure 2 on page 21, an occurrence of IMS segment SEGC and SEGD are mapped to a row of the DB2 table TABC/D. In the table, KEYA is the key of parent SEGA. KEYC, F4, and F5 are the key and non-key fields from SEGC. F6 and F7 are non-key fields from the immediately subordinate segment SEGD, which occurs only once.

If propagation is from DB2 to IMS, the arrows in Figure 2 on page 21 would simply be reversed.

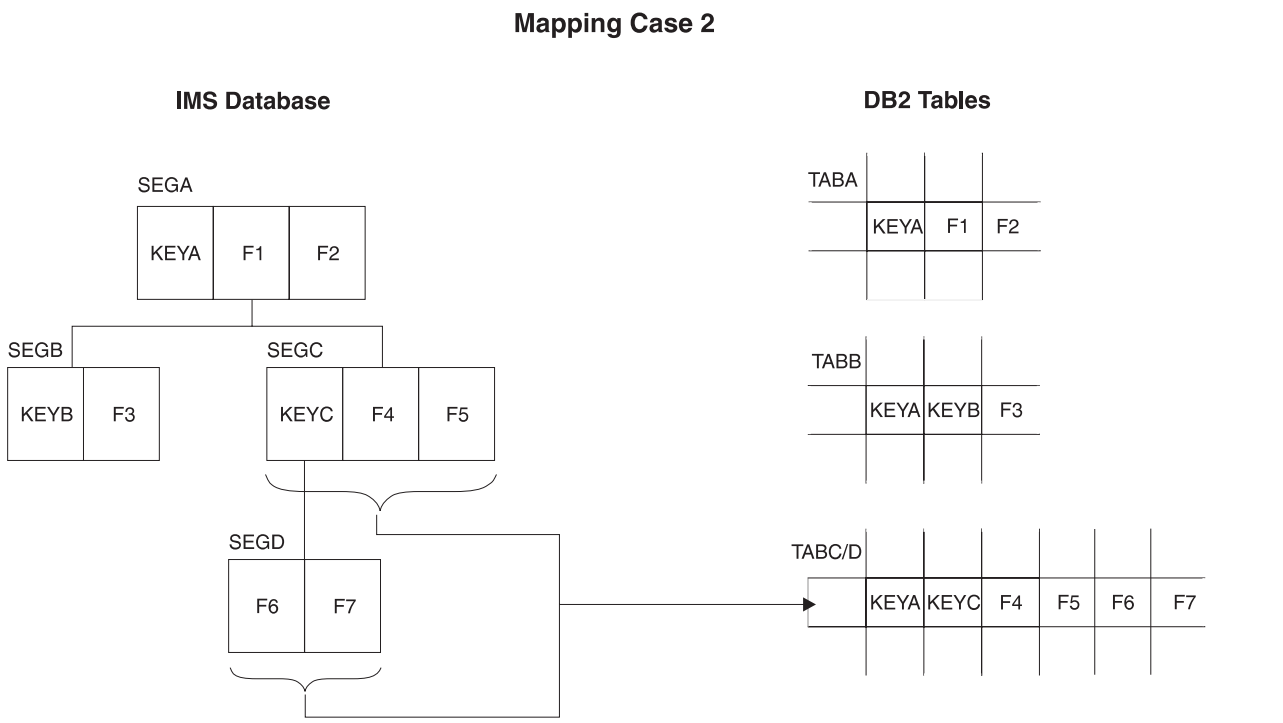


Figure 2. Mapping Case 2

Optional:

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

See "Mapping Options: Generalized Mapping Cases Only" on page 30 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 propagation and PRTYPE=E
- DB2-to-IMS 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.

Extension Segment for DB2-to-IMS Propagation and PRTYPE=E

For DB2-to-IMS synchronous propagation and for PRTYPE=E propagation requests, observe the following rules for extension segments:

1. Extension segments cannot have an IMS key field.
2. Dependents of extension segments cannot be propagated by PRTYPE=E (and the extension segment cannot contain internal segments or structures propagated by PRTYPE=E). If you need to propagate dependents of an extension segment, investigate propagating the extension segment to a table of its own through a mapping case 1 propagation request, instead of through a mapping case 2 propagation request.

DB2-to-IMS Propagation to Extension Segments

When you do DB2-to-IMS propagation, IMS DPROP generalized mapping inserts, deletes, and replaces extension segments under certain circumstances.

When you insert or update a row, the columns mapped by an extension segment might contain a default value, a value other than the default value, or a DB2 null value. Depending on the combination of all columns mapped to an extension segment, IMS DPROP decides how to propagate your SQL row based on whether it is an:

- SQL insert
- SQL update
- SQL delete

Synchronous Propagation of an SQL Insert: IMS DPROP determines how to propagate the SQL insert to:

Entity segment
The insert of the SQL row results in an IMS insert of the entity segment.

Extension segments
If at least one column mapped to an extension segment does not have a null value (for permitting columns) or the default,[3] for columns defined with NOT NULL WITH DEFAULT, then IMS DPROP inserts the extension segment.

If all columns mapped to the extension segment have either the null or the default value, then IMS DPROP looks at the DEFVEXT (default value extension) option:

- If DEFVEXT=N, the extension segment is not inserted
- If DEFVEXT=Y, the extension segment is inserted

Fields of the segment that are defined to IMS DPROP but not mapped are initialized with a default value corresponding to their data type[4]. Fields that are not defined to IMS DPROP are initialized with binary zeroes.

DB2-to-IMS propagation of variable-length segments is described in "Propagating Variable-Length Segments (DB2-to-IMS)" on page 46.

Propagation of an SQL Update: IMS DPROP determines which columns of the row have changed. Based on the mapping definitions, IMS DPROP also determines which segment types have at least one field mapped from a changed column.

---

3. For date, time, and timestamp columns, IMS DPROP does not distinguish between the default and nondefault values. Therefore, when processing date, time, and timestamp columns that do not have a null value, IMS DPROP assumes they have a nondefault value.

4. If you create propagation requests with a DXT or DataRefresher release before V2 R5, IMS DPROP initializes nonpropagated fields with binary zeros.

Segment types that have not been affected by the SQL update are not changed by IMS DPROP. The following rules apply to segments affected by the SQL update:

Entity segment
The entity segment is replaced. Mapped fields get their value from the columns of the modified row. Fields that are not mapped remain the same except when the REPL operation increases the length of the variable-length segment. IMS DPROP might need to assign initial values to some unmapped fields.

Extension segments
If no columns mapped to the extension segment have a null value (if permitted) or the default value (if defined with NOT NULL WITH DEFAULT), then IMS DPROP either replaces the existing extension segment (IMS REPL) or inserts a new extension segment (IMS ISRT).

- For a REPL, mapped fields get their value from the modified row. Fields that are not mapped remain the same except when the REPL operation increases the length of a variable-length segment. IMS DPROP might assign initial values to some unmapped fields.
- For an ISRT, all mapped fields get their value from the modified row. Fields defined to IMS DPROP but not mapped are initialized with a default value corresponding to their data types. IMS fields that are not defined to IMS DPROP are initialized with binary zeros.

If, after the update, all columns mapped to the extension segment have either a null value (if permitted) or the default, (if defined with NOT NULL WITH DEFAULT), then IMS DPROP looks at the DEFVEXT option you specified for the propagation request:

- If DEFVEXT=N, the extension segment is deleted if it existed before the SQL update. The extension segment is deleted even if it contains fields that are not mapped.
- If DEFVEXT=Y, the extension segment is either replaced (if it existed before the SQL update) or inserted. The field value is null, the default, or is determined by IMS REPL or IMS ISRT.

See "Propagating Variable-Length Segments (DB2-to-IMS)" on page 46 for a description of DB2-to-IMS propagation of variable-length segments.

Propagation of an SQL Delete: When a row is deleted, IMS DPROP deletes the entity and extension segments.

Mapping Case 3

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

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

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

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

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

In Figure 3 on page 25, mapping case 3 maps embedded structures. Each occurrence of an embedded structure is propagated together with the keys of the physical parent and ancestors up to the root. In Figure 3 on page 25, IMS segment SEGB contains two embedded structures, C and D. C and D occur multiple times within SEGB.

Figure 3 on page 25 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 3 on page 25 would simply be reversed.

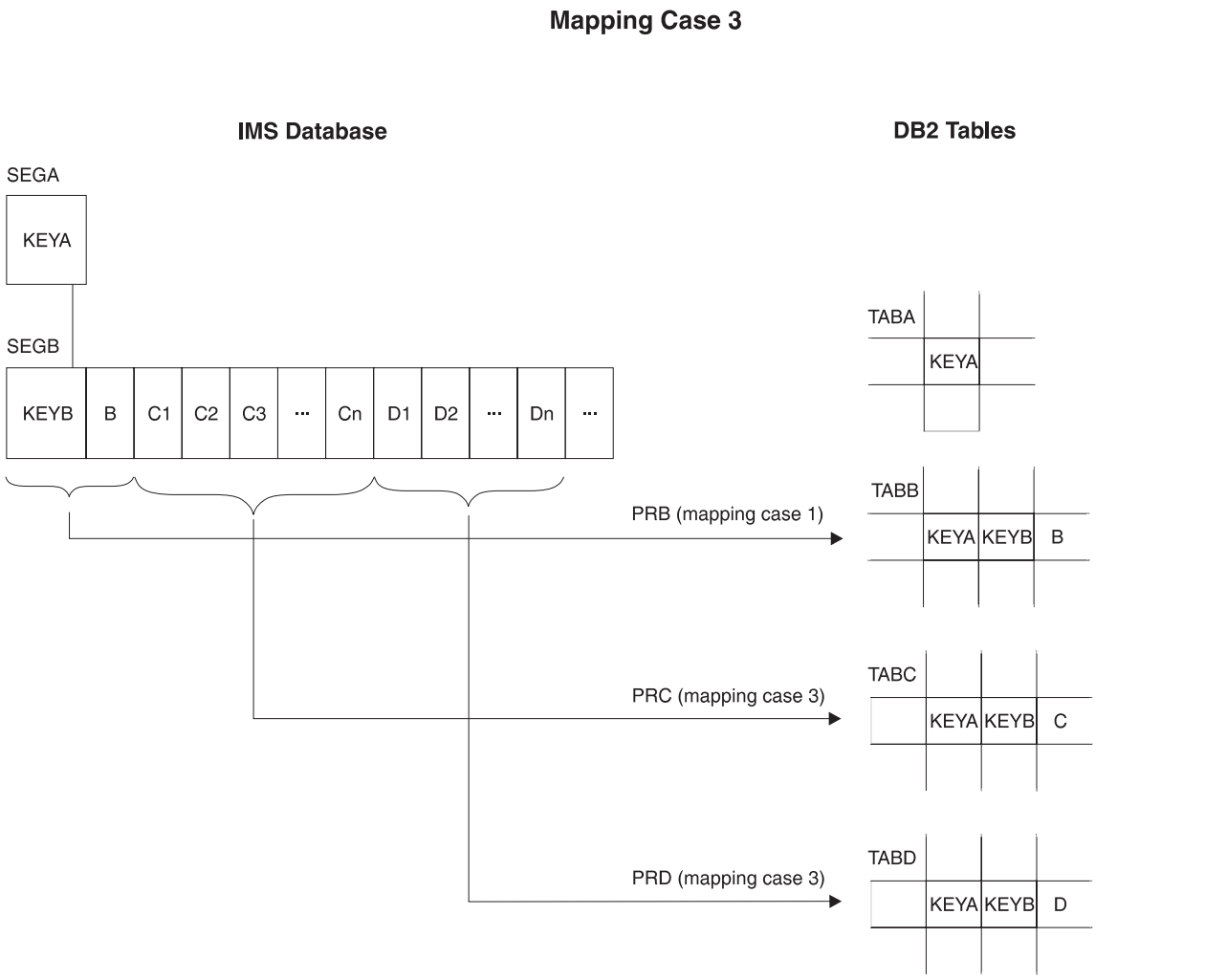


Figure 3. Mapping Case 3

For a mapping case 3 propagation request, you can use PATH data to include non-key fields from the IMS segment containing the embedded structure and from each segment in its physical path up to the root. For a detailed description of PATH data, see "Mapping Options: Generalized Mapping Cases Only" on page 30.

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

The following sections discuss mapping case 3 in association with:

- Definitions: Containing and Internal Segments
- Mapping Design for Mapping Case 3
- Internal Segments and Segment Exit Routines
- Unique Identification of Internal Segments
- Fixed/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

Definitions: Containing and Internal Segments

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

- Propagation involves embedded structures, a concept not defined in IMS. In IMS DPROP and DataRefresher, each embedded structure is called an *internal segment type*; see Figure 4 on page 26.
- The IMS segment itself is called the *containing IMS segment* in IMS DPROP and DataRefresher.
- IMS DPROP views the containing IMS segment as the parent of the internal segments.
- IMS DPROP considers the internal segment to be the entity segment.

Containing Segment and Internal Segment Type

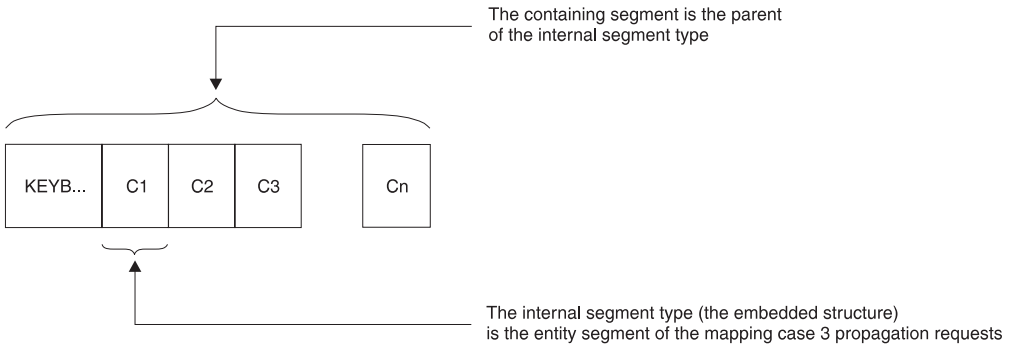


Figure 4. Containing Segment and Internal Segment Type

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

Generalized mapping of IMS DPROP 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 50.
- Use different names for different internal segment types.

Mapping Design for Mapping Case 3

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

The mapping is a two-step process, as illustrated in Figure 5 on page 27.

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

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

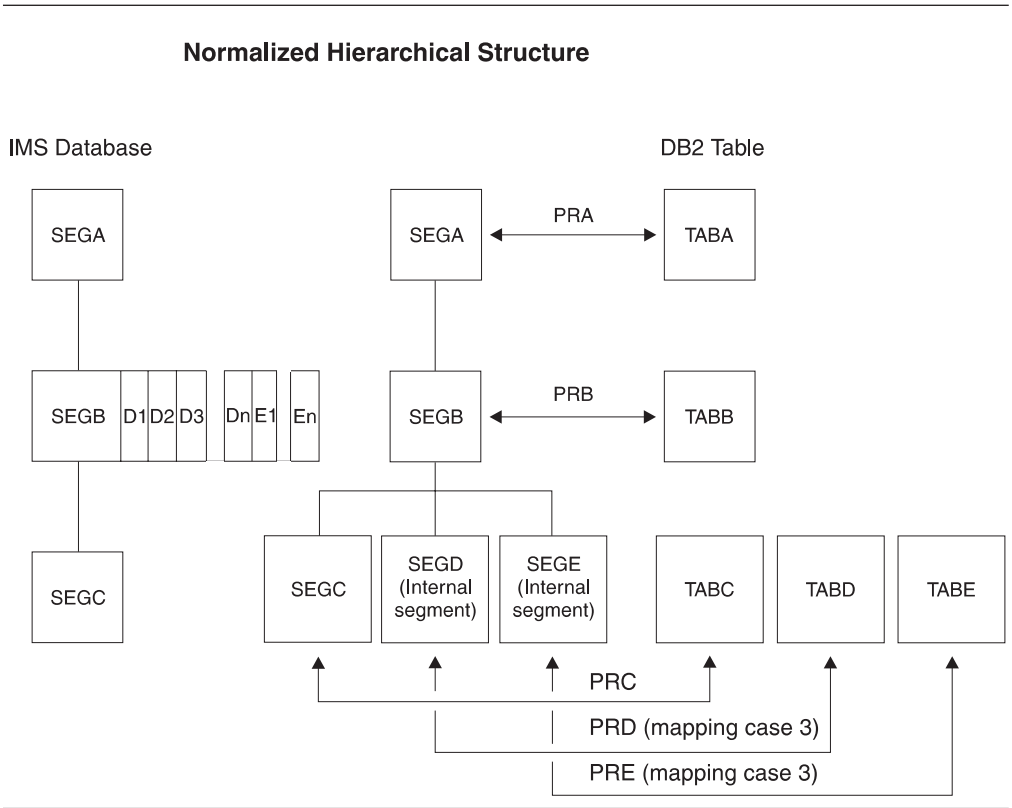


Figure 5. Conceptually Normalizing the Database for Mapping Case 3

Figure 5 helps illustrate two steps:

Step 1 Conceptually transform the IMS database structure into a normalized hierarchical structure.

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

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

Step 2 Using propagation requests, map the normalized hierarchical structure into DB2. Internal segments SEGD and SEGE are mapped with mapping case 3. Other segments are mapped by propagation requests belonging to a different mapping case.

Internal Segments and Segment Exit Routines

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

You might need to write a Segment exit routine for mapping case 3 if you 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 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.
- Segment exit routines are easier to write than Propagation exit routines. You do not need to provide SQL logic. When doing DB2-to-IMS propagation, you also do not need to provide DL/I logic.
- You can use the trace facility of IMS DPROP 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 59 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, the RUP compares the before and after image of the containing segment. By comparing the ID fields of internal segments, the RUP determines which occurrences of internal segments have been inserted, replaced, and deleted. The 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 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 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.

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

DB2-to-IMS Propagation of Internal Segments

If you propagate an internal segment from DB2-to-IMS, then you also need to propagate the containing IMS segment.

Do not define the same bytes in the IMS segment as both part of the containing and internal segment. Also do not define them as part of two different internal segments. You can map the key and PATH data of the containing segment to the target of both the containing and internal segment.

Handle the containing segment-internal segment relationship as a parent-child segment when setting up matching RIRs.

DLU Processing of Internal Segments

If DLU processes an internal segment propagated with a PRTYPE=E, then the containing IMS segment must also be propagated with a PRTYPE=E.

SEG= Keyword on IMS DPROP Control Statements

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

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

User Mapping Cases

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

Mapping Options: Generalized Mapping Cases Only

You can use two mapping options with generalized mapping cases:

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

PATH Data

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

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

This section discusses:

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

Figure 6 on page 31 shows propagation of PATH data. In the figure, PR3, which propagates entity segment SEG3 to TAB3, includes the following in its mapping:

- Key fields from each segment in the physical hierarchical path, from the root down to entity segment SEG3. These fields are KEY1, KEY2, and KEY3.
- PATH data fields from segments in the physical hierarchical path, from the root down to physical parent SEG2. These fields are DATA1 and DATA2.
- Non-key fields from the entity segment. These fields are DATA3 and other fields not shown in Figure 6 on page 31.

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

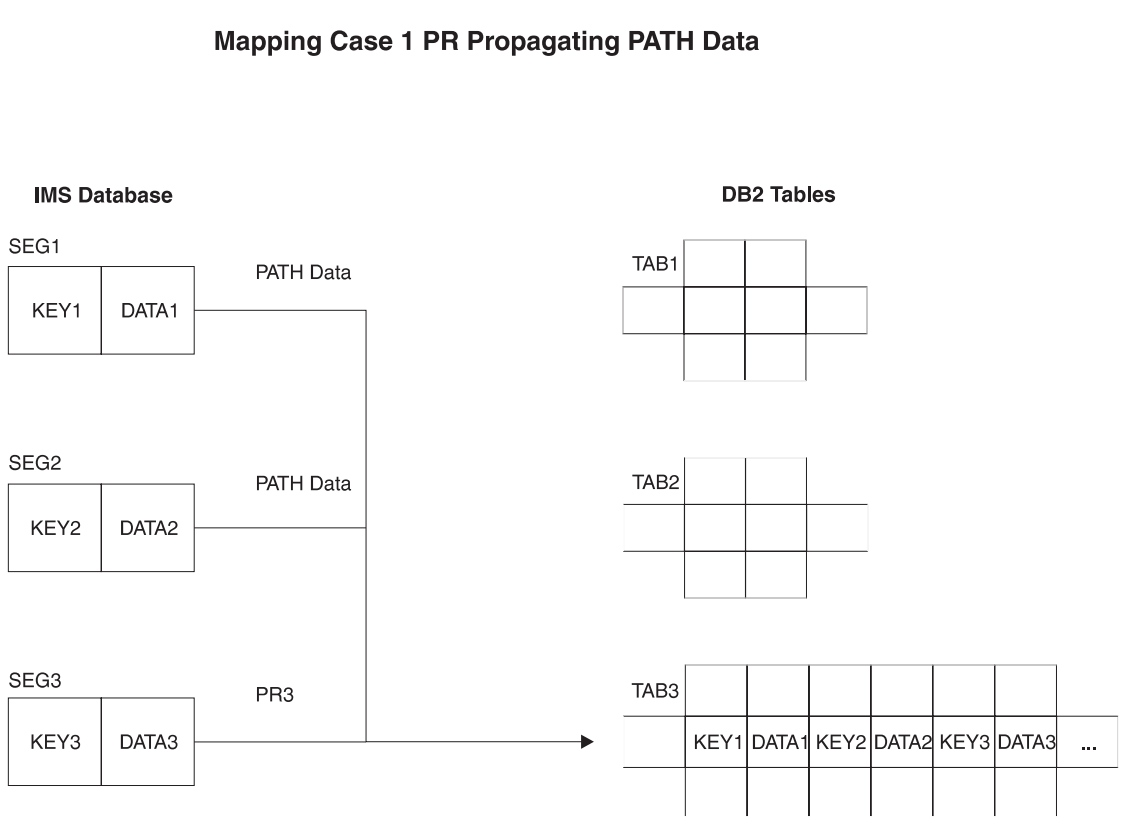


Figure 6. Mapping Case 1 Propagation Request Propagating PATH Data

Uses of PATH Data

IMS DPROP supports two different uses of PATH data. Because the 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 83). If you use the DB2 copy of your data only in read-only mode (for example, for decision support purposes) you can intentionally denormalize it to increase performance of your DB2 queries. Intentional denormalization often avoids the performance overhead of joins during queries.

The example in Figure 7 on page 33 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 7 on page 33 illustrates denormalization of data. In the figure, PR2 belongs to mapping case 1 and propagates the entity segment SKILLS to the table SKILLS. PR2 includes in its mapping as PATH data the NAME field from the physical parent.

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

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

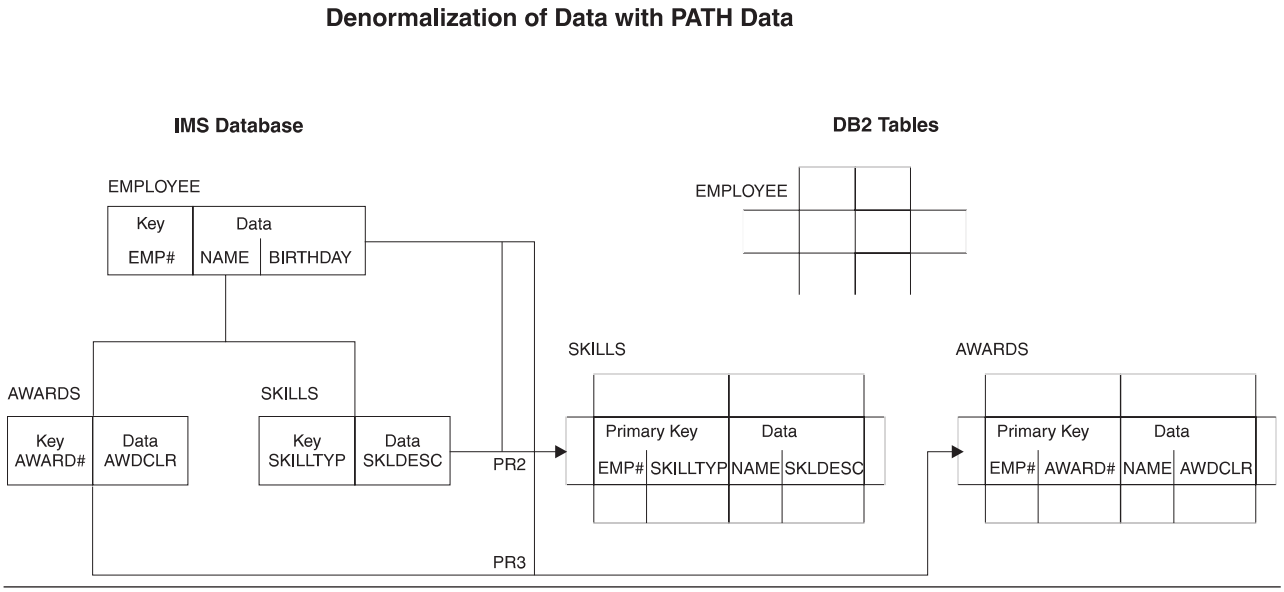


Figure 7. Denormalization of Data with PATH Data

Rules for PATH=DENORM

1. A propagation request can be defined only as PRTYPE=L. Only one-way IMS-to-DB2 propagation is supported.
2. Fields included in PATH data can change their values. Your IMS application programs can change the value of PATH data with REPL calls. If the value of PATH data can change, make sure 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 53, 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[5] mapped by the propagation request and located in either:
      - The parent/ancestor
      - A segment higher in the hierarchy

      Figure 8 on page 35 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:

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

- 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.

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

In the example in Figure 8 on page 35, 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.

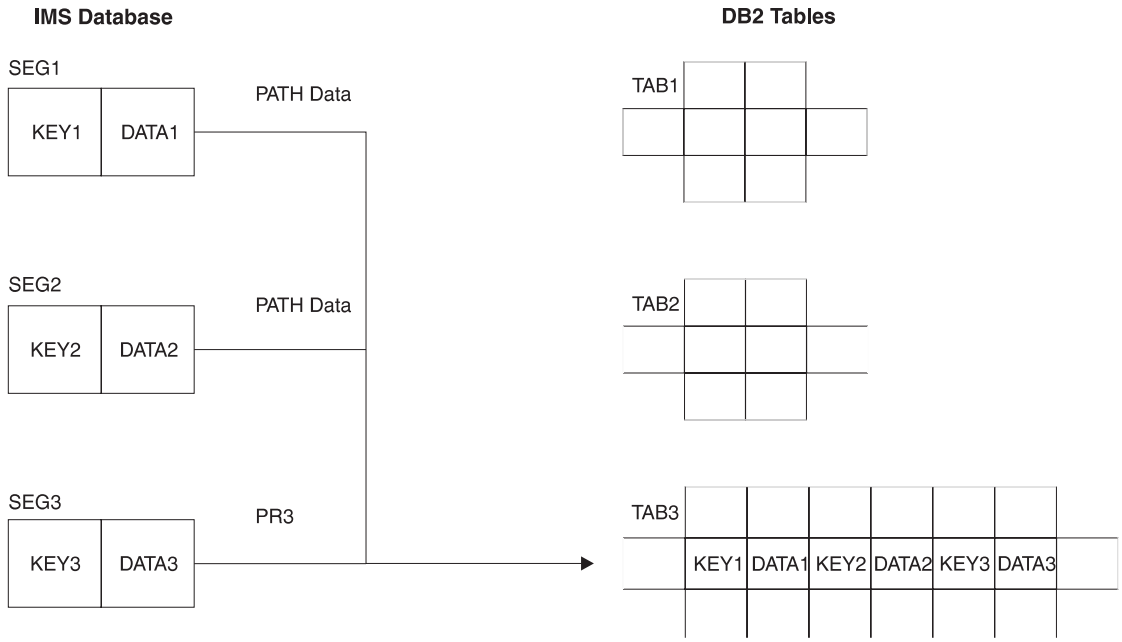


Figure 8. Identifying Parent/Ancestors Contributing Modifiable PATH Data to PR3

PATH=ID: Mapping ID Fields of a Physical Parent/Ancestor

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

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

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

In the example shown in Figure 9 on page 37, 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 62.

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

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

   In Figure 9 on page 37, all propagation requests propagating SEG2 and all propagation requests propagating its dependent SEG3 must map ID field ID2 of SEG2 to the DB2 primary key of their respective tables.

In Figure 9 on page 37, 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.

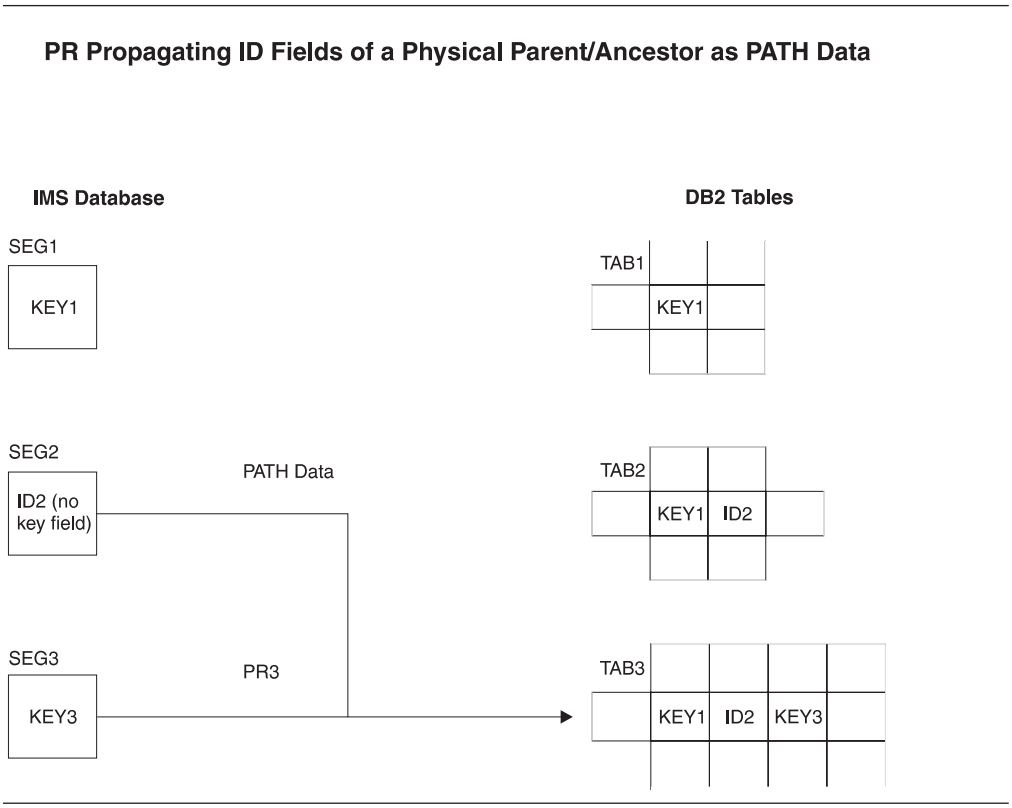


Figure 9. PR Propagating ID Fields of a Physical Parent/Ancestor as PATH Data

WHERE Clause

You can specify a WHERE clause in the propagation request for mapping cases 1 and 2, but not 3. The WHERE clause specifies under which conditions a segment occurrence is to be propagated from IMS to DB2. The conditions are based on field values or a combination of field values. 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 38.

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

Figure 10 on page 38 shows mapping with a WHERE clause.

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

In Figure 10 on page 38, segment SEG2 contains redefined data. In this example, SEG2 can contain two different kinds of data. The kind of data in a particular occurrence of SEG2 is identified by the value of field F2, which can have the following values: 'A', 'a', 'B', and 'b'.

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

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

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

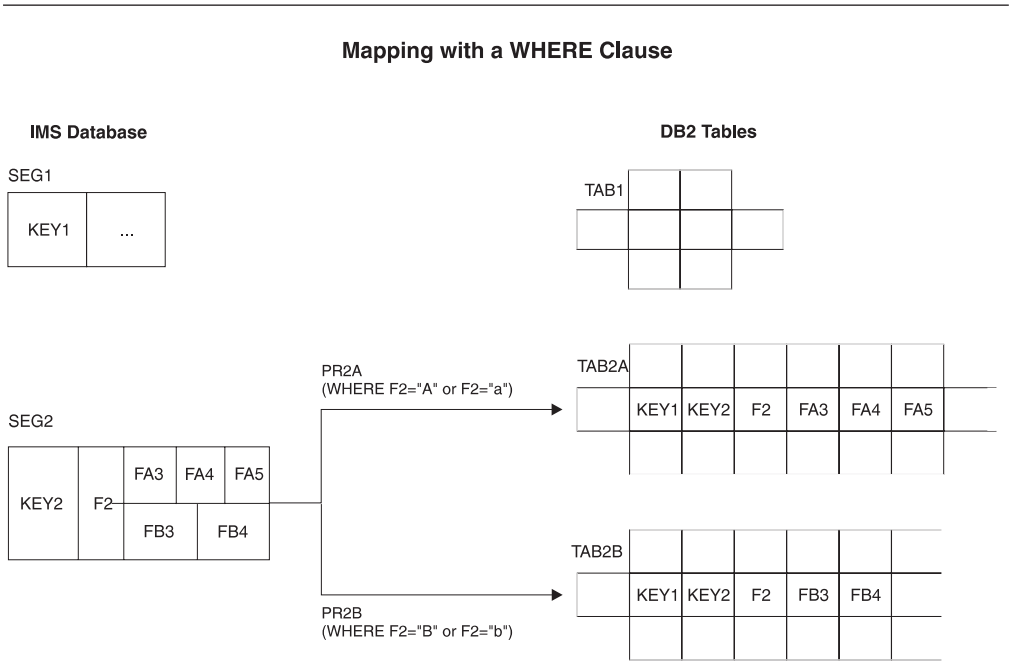


Figure 10. Mapping with a WHERE Clause

The following sections discuss:
- Selective Propagation
- Fields That Can Be Included in the WHERE Clause
- Fields That Cannot Be Included in the WHERE Clause
- Conditions and Operators Used with the WHERE Clause
- 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 propagation. However, IMS DPROP support of the WHERE clause for DB2-to-IMS 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 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 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 That Can Be Included in the WHERE Clause

In the WHERE clause, you can include:

- IMS *key fields* or subfields of keys from each segment in the physical hierarchical path of the entity segment, from the entity segment up to the root.
- Fields belonging to *PATH data* in the parent or in an ancestor of the entity segment, when the propagation request specifies PATH=ID. These are fields that cannot change their value. IMS DPROP 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[6].
    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 Cannot Be Included in the WHERE Clause

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

- Fields in dependents of the entity segment (for example, fields located in an extension segment of mapping case 2). IMS DPROP checks for this when you define a propagation request.
- PATH data fields if the propagation request specifies PATH=DENORM, meaning the PATH data fields can change their value. IMS DPROP checks for this when you define a propagation request.
- For mapping case 2, fields located in the entity segment that can change their value[7]. IMS DPROP checks for this when you update the entity segment or target row.

---

6. 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.
7. MVG writes warnings when it detects a non-key field of a mapping case 2 entity in a WHERE clause.

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

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

Conditions and Operators Used with the WHERE Clause

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

AND operator
OR operator

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

IMS DPROP and DataRefresher support comparisons using the following operators:

=
>
>=
<
<=
¬=

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

NOT
LIKE
NOT LIKE
IN
NOT IN
BETWEEN

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

Recommendations for Propagating Parent Segments with a WHERE Clause

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

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

Recommendation for Propagating Logical Parent Segments with a WHERE Clause

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

You cannot include either:

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

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

Chapter 3. 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 done 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:

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

DB2-to-IMS Limitations

IMS DPROP limitations for DB2-to-IMS propagation are:

- For segments having no unique IMS key field and an IMS insert rule of HERE, propagation of SQL inserts is done as if the insert rule is FIRST.
- When using the DLU, IMS DPROP does not necessarily load segments without a unique IMS key field in the sequence expected by your application programs. Refer to the *IMS DataPropagator Reference* for more information about the DLU.
- During propagation of DEDB direct-dependent segments, IMS DPROP does not modifies or sets subset pointers.

IMS Logical Relationship Rules

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

Paired Logical Children

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

  If you do not observe these rules for generalized mapping cases, IMS DPROP either writes warning messages when creating the propagation request (PRTYPE=L and F) or does not create the propagation request (PRTYPE=E). (For an explanation of types of propagation requests, see "Selecting a Propagation Request Type" on page 12.)
- When using the DLU, if the pairing is physical, the DLU recreates from the DB2 tables *only* that segment type of the pair that is to be propagated. You should give DLU the other segment type of the pair as complementary data. For more information on this subject, see "Considerations for Paired Segment Types" on page 171.

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
- Unidirectional relationships, IMS DPROP generalized mapping logic usually requires a delete rule of PHYSICAL

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

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

  Retrieving the logical parent segment or its ancestor through a physical path should provide the same results as accessing a DB2 row.
- You cannot establish a matching DB2 referential integrity relationship between the target table of the logical parent and the target of the logical child. If you implement a DB2 referential integrity relationship between the targets of a logical parent and logical child, propagation usually fails.

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

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

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

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

Requirement for a DB2 Primary Key

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

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 11.

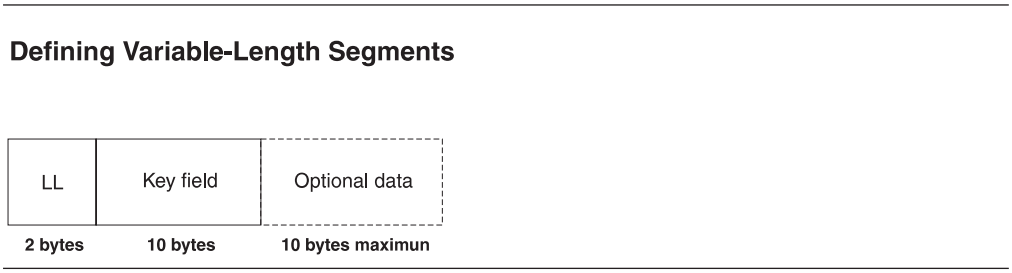


Figure 11. Defining Variable-Length Segments

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

The application program that manipulates this segment can insert or replace a segment occurrence so the data field is either:
- Completely contained within the segment occurrence (the segment length being 22 bytes)
- Completely absent (the segment length being 12 bytes)
- Shorter than the 10 bytes expected by IMS DPROP (the segment being greater than 12 bytes and less than 22 bytes), which is not valid for IMS DPROP

Propagating Variable-Length Segments (DB2-to-IMS)

When propagating an SQL insert or update to a variable-length segment, IMS DPROP's generalized mapping needs to determine and set the length of the segment.

IMS DPROP uses the following rules to determine segment length:
1. For *insertion* of a segment, IMS DPROP uses the right-most *propagated* field that is not synchronously propagated from a DB2 null value.
2. For *replacement* of a segment, IMS DPROP uses the right-most of the following two fields:
   - The right-most *propagated* field that is not synchronously propagated from a DB2 null value
   - The right-most *nonpropagated* byte that exists in the before image of the IMS segment occurrence

For DEDB segments, if the resulting segment length is smaller than the minimum bytes limit specified in the IMS DBD, IMS DPROP increases the segment length to agree with the minimum bytes specification. The segment length is increased to either the minimum bytes value— if this does not result in fields only partially contained in the segment—or to the next higher value that does not result in partial fields.

These rules sometimes cause segment occurrences that are longer than you want. For example, when most propagated fields are mapped from DB2 columns defined with NOT NULL WITH DEFAULT, these IMS DPROP rules often cause long segment occurrences with trailing fields containing default blank and zero values.

To prevent trailing fields containing blanks and zero values, consider using a Segment exit routine. A Segment exit routine can set the length of the IMS segment to your requirements. For example, the Segment exit routine can reduce segment length by eliminating trailing fields containing blanks and zeroes.

Propagating a Subset of Fields or Columns

With synchronous propagation, the simplest scenario is when all fields in a segment and all columns in a table are propagated. However, in some cases, you might not want to propagate all fields in a segment or all columns in a table.

The following sections discuss:
- Propagation of a Subset of Fields in a Segment
- Propagating a subset of columns in a table

Propagation of a Subset of Fields in a Segment

Propagating a subset of the fields in a segment is straightforward for one-way IMS-to-DB2 propagation. However, for DB2-to-IMS propagation, during insert operations, IMS DPROP sets the value of nonpropagated fields to an initial value.

This section describes:
- How nonpropagated fields are handled by IMS DPROP during DB2-to-IMS propagation
- Considerations and recommendations for one-way DB2-to-IMS and two-way propagation

Assigning Values to Nonpropagated Fields During DB2-to-IMS Propagation:
When doing DB2-to-IMS propagation, IMS DPROP distinguishes between nonpropagated fields that have been explicitly defined to IMS DPROP and nonpropagated fields that have not been defined. For propagation requests defined with DataRefresher, IMS DPROP can make a distinction only if you use DXT V2 R5 or DataRefresher. Explicitly defining nonpropagated fields has the advantage that when a segment is inserted during DB2-to-IMS propagation, the fields are set to the initial/default value for the data type of the field.[8] Nonpropagated fields that have not been defined are set to binary zeroes.

During DB2-to-IMS propagation, IMS DPROP does the following for fields that are not propagated:
- When *replacing* a segment, IMS DPROP does not change the value of nonpropagated fields. If a replace increases the length of a variable-length segment, IMS DPROP sets nonpropagated fields to an initial value.

8. If you create propagation requests with a DXT release before V2 R5, IMS DPROP initializes nonpropagated fields with binary zeroes.

- When *inserting* a segment, IMS DPROP sets nonpropagated fields to either:
  - The default value for the data type of the field,[9] if the field has been defined to IMS DPROP
  - Binary zeroes, if the field has not been defined to IMS DPROP

  Unless you provide complementary input data, DLU also sets and uses the value of nonpropagated fields when creating or re-creating the IMS database. If you do provide complementary input, for example a previous copy of the IMS database, DLU sets the value of the nonpropagated fields from the content of the complementary input data. For more information on this subject, see "How the DLU Selects and Processes Input Data" on page 167.

Recommendation for One-Way DB2-to-IMS and Two-Way Propagation: When doing one-way DB2-to-IMS or two-way propagation, we recommend that you map and propagate all fields in a segment, (with the exception of"filler" fields and unused fields) to make it easier to recreate the IMS data copy based on the DB2 data copy:

- If you recreate the IMS database with the DLU, DLU sets nonpropagated fields to either an initial value or to the value in the previous copy of the IMS database that you provide as complementary input. Unless all nonpropagated fields are unused fields, you must provide DL/I update programs to set the real value of these fields.
- If you use CCU-generated DL/I repair statements, then repair segments inserting IMS segments set nonpropagated fields to an initial value. Unless all nonpropagated fields are unused fillers, you must provide DL/I update programs to set the real value of these fields.

Additional Considerations for One-Way DB2-to-IMS Propagation:
Nonpropagated fields can only be updated by DL/I calls in your updating IMS programs.

Make sure your IMS programs do not update propagated fields and do not delete or insert the whole segment. With one-way DB2-to-IMS propagation, IMS program updates, deletes, and inserts are not propagated to DB2 and, therefore, jeopardize the consistency between the DB2 and IMS copy. You can use the IMS DBDGEN EXIT= keyword to specify that RUP is to be invoked. The RUP catches updates, deletes, and inserts. If the RUP finds inconsistencies, it initiates propagation failure.

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

This section describes:

- Assigning Values to Nonpropagated Columns
- Requirements for one-way IMS-to-DB2 propagation and two-way propagation

Assigning Values to Nonpropagated Columns During IMS-to-DB2 Propagation:
During IMS-to-DB2 propagation:

9. If you create propagation requests with a DXT release before V2 R5, IMS DPROP initializes nonpropagated fields with binary zeroes.

- 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 for One-Way IMS-to-DB2 Propagation: and Two-Way 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 or two-way propagation, map all columns in a propagated table to make it easier to recreate the DB2 data copy based on the IMS data copy:

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

Additional Considerations for One-Way IMS-to-DB2 Propagation:
Nonpropagated columns can be updated only by your updating SQL statements.

Make sure your SQL statements do not update propagated columns and do not delete or insert the whole row. With one-way IMS-to-DB2 propagation, SQL updates are not propagated to the IMS copy and, therefore, jeopardize consistency between the DB2 and IMS copy. Consider using DB2 security to prevent SQL statements from updating propagated columns or inserting and deleting rows. See "Updates to Nonpropagated Columns" on page 144 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 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 or from the Same Table

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

Propagating One Segment to or from Multiple Tables

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

PRTYPE=L and One-Way IMS-to-DB2 Propagation

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 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.

All PRTYPE=Es propagating a group of logically related IMS databases and all PRTYPE=Es propagating the tables of one DB2 referential integrity structure should propagate in the same direction.

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

PRTYPE=U

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

Using Propagation Request Sets

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

Sometimes it is convenient to group logically related propagation requests into the same propagation request set.[10] 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

10. 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.

you use the PRSET= keyword, IMS DPROP applies the control statement to the propagation requests belonging to the specified propagation request set. You might find it more convenient to specify a PRSET ID on IMS DPROP control statements than to specify a long list of individual propagation request IDs.

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

Examples of Propagation Request Set Use

Examples of using propagation request sets are:

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

You can use:

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

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

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

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

Defining Propagation Requests with Qualified or Unqualified Table Names

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

Qualified Table Names

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

IMS-to-DB2 Propagation: During propagation request definition, if you specify a qualified table name, the SQL statements in the SQL update module that has been generated use the specified qualified table name. Therefore, the propagation request propagates to the same qualified table name; it cannot propagate to other identically structured tables with other table name qualifiers.

DB2-to-IMS Propagation: The propagation request propagates only updates from the table whose qualified name is defined in the propagation request.

Unqualified Table Names

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

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

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

When binding a package or the plan of the propagating application, the BIND process sets the qualifier, which determines the qualified table name of the propagated table. If you are using the bind package function, use the QUALIFIER= keyword of the BIND PACKAGE command to set the qualifier. If you are not using the bind package function, setting the qualifier is more complex; for more information on this subject, refer to "DB2 ALIAS and SYNONYM Statements" on page 156.

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-to-IMS Propagation: If, within a single IMS schedule, an application program tries to update two different tables that have the same name but different table qualifiers and that are being propagated by the same unqualified propagation request, IMS DPROP propagates the update to the first table, but backs out the update to the second table and issues error message EKYH405E.

SCU Considerations: Usually you do not use the SCU to deactivate or suspend propagation requests with unqualified table names because the SCU deactivates or suspends propagation to or from multiple tables. You cannot deactivate or suspend propagation of the propagation request to or from *one* of the multiple propagated tables. For this reason, propagation requests with unqualified table names might be less useful in a production environment.

Also the SCU waits until the updates to *all* tables having the same unqualified table name and defined for changed data capture have been quiesced when processing an ACTIVATE, DEACTIVATE, or SUSPEND control statement.

DB2 Referential Integrity Guidelines

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

If you are using a generalized mapping case, there are rules for implementing DB2 RIRs involving propagated tables. By following these rules, you avoid failures in the propagation:

- IMS updates (IMS-to-DB2)
- DB2 updates (DB2-to-IMS)

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

Observe the following rules when implementing RIRs for propagated tables.

1. For one-way IMS-to-DB2 propagation, implementation of RIRs involving propagated tables is *optional*. If implemented, the DB2 parent/child RIRs should be a matching *subset* of the IMS parent/child relationships.

   If RIRs are not a matching subset, propagation fails. The DB2 referential integrity requirements do not match any equivalent IMS parent/child relationship.
2. For one-way DB2-to-IMS propagation, implementation of RIRs involving propagated tables is strongly *recommended*. The implemented DB2 parent/child RIRs should be a *superset* of the IMS parent/child relationships. You should have least one DB2 parent/child RIR matching each IMS parent/child relationship or propagation fails. You can implement, in addition to the matching RIRs, additional DB2 parent/child RIRs.
3. For two-way propagation, implementation of RIRs involving propagated tables is strongly *recommended*.

   Use the rules for both one-way IMS-to-DB2 propagation and one-way DB2-to-IMS propagation. The implemented DB2 parent/child RIRs should be both a *subset* and a *superset* of the IMS parent/child relationships. You should have a one-to-one correspondence between each DB2 parent/child RIR and IMS parent/child relationship.

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

However, if you plan to start with one-way IMS-to-DB2 propagation and later switch to one-way DB2-to-IMS or two-way propagation you might want to initially implement DB2 RIRs using the rules for two-way propagation to get experience with the DB2 RIRs recommended for successful two-way and one-way DB2-to-IMS propagation.

The following sections discuss:

- Defining DB2 RIRs to match IMS relationships
- Using DB2 delete rules for matching RIRs
- Defining DB2 RIRs for one-way IMS-to-DB2 propagation
- Defining DB2 RIRs for one-way DB2-to-IMS propagation
- Defining DB2 RIRs for two-way propagation
- Implementing non-matching RIRs for one-way IMS-to-DB2 and two-way propagation

Defining DB2 RIRs to Match IMS Relationships

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

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

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

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

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

   Exception to the ideal case are:

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

     The conceptual fully concatenated key is defined in "Terminology Related to Keys" on page 59.
   - For PRTYPE=L, the rules less strict than for PRTYPE=E. For a DB2 RIR matching a *physical* IMS parent/child relationship: the primary key of the parent table and the foreign key of the child table should be mapped from the same combination of fields located in the:
     - IMS fully concatenated key of the physical parent segment.
     - Data portion of the physical parent or physical ancestor segment. These fields should not change their value.
   - For a DB2 RIR matching a *logical* IMS parent/child relationship: the primary key of the parent table and the foreign key of the child table should be mapped from the same combination of fields located in the IMS fully concatenated key of the logical parent segment.

Using DB2 Delete Rules for Matching RIRs

This section describes DB2 delete rules you can use to implement RIRs matching IMS parent child/relationships. Valid /B2 delete rules for RIRs matching IMS *physical* parent/child relationships are different from those matching IMS *logical* parent/child relationships.

RIR Matching a Physical IMS Parent/Child Relationship

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

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

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

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

RIR Matching a Logical IMS Parent/Child Relationship

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

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

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

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

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

Table 4. Valid Combinations of Delete Rules for DB2 RIRs Matching IMS Logical Parent/Child Relationships (Generalized Mapping Cases)

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

Table 4. Valid Combinations of Delete Rules for DB2 RIRs Matching IMS Logical Parent/Child Relationships (Generalized Mapping Cases) (continued)

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

Defining DB2 RIRs for One-Way IMS-to-DB2 Propagation

For one-way IMS-to-DB2 propagation, implementation of DB2 RIRs is optional. To avoid propagation failures, DB2 RIRs should be a matching subset of the IMS parent/child relationships. Specifically, you can implement:

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

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

Defining DB2 RIRs for One-Way DB2-to-IMS Propagation

For one-way DB2-to-IMS propagation, implementation of DB2 RIRs is strongly recommended. To avoid propagation failures, DB2 RIRs should be a matching superset of the IMS parent/child relationships. Specifically:

- The physical parent, the logical parent, and the ancestors of a propagated child segment must also be propagated using one-way DB2-to-IMS propagation.
- For each propagated entity child segment, you should implement a DB2 parent/child RIR that matches the physical IMS parent/child relationship.

For each propagated logical child segment, you should also implement a DB2 parent/child RIR that matches the logical IMS parent/child relationship.

- You can implement additional DB2 RIRs.

Defining DB2 RIRs for Two-Way Propagation

For two-way propagation, implementation of DB2 RIRs is strongly recommended. To avoid propagation failures, DB2 RIRs between propagated tables should match IMS parent/child relationships. Specifically:

- The physical parent, the logical parent, and the ancestors of a propagated child segment should also be propagated through two-way propagation.
- For each synchronously-propagated entity child segment, you should implement a DB2 parent/child RIR that matches the physical IMS parent/child relationship.

  For each propagated logical child segment, you should also implement a DB2 parent/child RIR that matches the logical IMS parent/child relationship.
- You should implement RIRs between a propagated parent table and a nonpropagated child table with a DB2 delete rule of ON DELETE CASCADE or ON DELETE SET NULL.

You must restrict the form and numbers of DB2 RIRs you implement to prevent propagation failures. If you choose to implementation other RIRs, you risk propagation failure.

Implementing Non-matching RIRs for One-Way IMS-to-DB2 and Two-Way Propagation

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

Defining Unique Indexes

To avoid propagation failures, observe IMS DPROP rules when defining:

- Unique DB2 indexes and doing IMS-to-DB2 propagation
- Unique IMS secondary indexes and doing DB2-to-IMS 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
- Truly unique IMS secondary indexes and one-way DB2-to-IMS propagation
- Unique indexes and two-way 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.

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

Truly Unique IMS Secondary Indexes and One-Way DB2-to-IMS Propagation

DB2-to-IMS propagation might fail if a truly unique IMS secondary index enforces uniqueness not enforced by DB2. In such situations, an SQL insert or update statement might not be successfully propagated because the update violates the uniqueness enforced by the IMS secondary index.

To avoid problems, do not implement truly unique IMS secondary indexes unless they correspond to unique DB2 secondary indexes. If you need a truly unique IMS secondary index that is not matched by an existing unique DB2 index, consider implementing the required matching, unique DB2 index.

One-way DB2-to-IMS propagation does not limit the number of unique DB2 indexes you can implement.

Attention: Even a truly unique IMS index that corresponds to a unique DB2 secondary index can cause synchronous propagation to fail if one SQL statement updates multiple rows. The sequence of the propagating updates to the individual segment occurrences is unpredictable and can result in duplicates in the IMS secondary index. If you want to eliminate the risk of such synchronous propagation failures, consider defining your IMS secondary indexes as non-unique.

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

Unique Indexes and Two-Way Synchronous Propagation

Two-way propagation could fail if a truly unique IMS secondary index enforces uniqueness not enforced by DB2 or if a DB2 index enforces uniqueness not enforced by IMS. In such situations, an IMS or SQL insert or update statement might not be successfully propagated because the update violates the uniqueness enforced by DB2 or IMS. Therefore, match unique DB2 indexes (other than the DB2 primary index) with truly unique IMS secondary indexes.

One SQL statement that updates multiple rows can sometimes cause synchronous propagation to fail. Therefore, avoid using truly unique IMS indexes for two-way propagation. Consider defining your IMS indexes as non-unique.

Key Mapping Rules by Propagation Request Type

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

- Terminology related to keys
- Overview of the key mapping rules
- Rules for PRTYPE=E (extended function)
- Rules for PRTYPE=L (limited function)
- Comparison of key mapping rules

Terminology Related to Keys

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

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

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

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

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

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

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

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

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

A logical child segment has two concatenated keys:

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

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

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

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

  See Figure 14 on page 72 for an illustration of concatenated keys.

ID fields
Ideally, a propagated entity segment has a unique IMS fully concatenated key. However, IMS DPROP's generalized mapping supports propagation when

segments do not meet this ideal. Identification (ID) fields and conceptual keys are useful when some of your segments:

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

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

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

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

Conceptual key

This term applies only to PRTYPE=E.

For segments that are not unique under their parent and do not have a unique IMS key but are uniquely identifiable using ID fields, the conceptual key is a non-overlapping combination of the non-unique IMS key field and ID fields. This combination must identify the segment uniquely under its parent.

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

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

Conceptual fully concatenated key

This term applies only to PRTYPE=E.

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

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

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

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

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

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

support segments with a unique conceptual fully concatenated key similar to segments with a unique IMS fully concatenated key.

Conceptual concatenated key
This term applies to only PRTYPE=E.

The conceptual concatenated key of a segment is the conceptual keys of the segment's immediate physical parent and physical ancestors. Unlike the conceptual *fully* concatenated key, the conceptual concatenated key does not include the conceptual key of the segment itself.

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

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

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

DB2 foreign key
DB2 foreign keys define DB2 RIRs.

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

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

IMS DPROP 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 IMS DPROP 1.1, you can still define PRTYPE=F in IMS DPROP 1.2 and 2.1.

Rules For PRTYPE=E (Extended Function)

PRTYPE=Es have strict key mapping rules but also provide more function than PRTYPE=L. Figure 12 on page 66 and Figure 13 on page 68 illustrate the key mapping rules for PRTYPE=E. Key mapping rules for PRTYPE=E are:

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

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

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

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

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

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

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

- The entity segment must have a *unique conceptual fully concatenated key*. The physical parent and all physical ancestors of the entity segment must also have a unique conceptual fully concatenated key. As explained in more detail on page 60, the conceptual fully concatenated key results from adding:
  - The IMS fully concatenated key
  - The ID fields of the entity segment its physical parent/ancestors

  ID fields are fields that contribute to uniquely identifying a segment.
- The DB2 primary key must be the propagated conceptual fully concatenated key of the entity segment. The DB2 primary key must be mapped by the conceptual fully concatenated key of the entity segment; and the conceptual fully concatenated key of the entity segment must be mapped to the DB2 primary key. The fields being mapped to the DB2 primary key must not overlap.

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

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

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

   In Figure 13 on page 68, 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 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 describes how the DB2 foreign key of a dependent table should be mapped when defining DB2 RIRs that match IMS parent/child relationships.

For one-way DB2-to-IMS and two-way propagation, you should implement matching DB2 RIRs. For one-way IMS-to-DB2 propagation, definition of matching DB2 RIRs is optional. See “DB2 Referential Integrity Guidelines” on page 53 for more information on this subject.

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

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

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

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

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

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

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

For more information on the CCU and direct verification, see "Overview of the CCU" on page 219 and the *Reference*.

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

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

Example of Mapping Keys in Ideal Case (PRTYPE=E)

Figure 12 on page 66 shows the ideal case for mapping keys with PRTYPE=E. All entity segments have a unique IMS fully concatenated key. The DB2 primary key is the propagated IMS fully concatenated key.

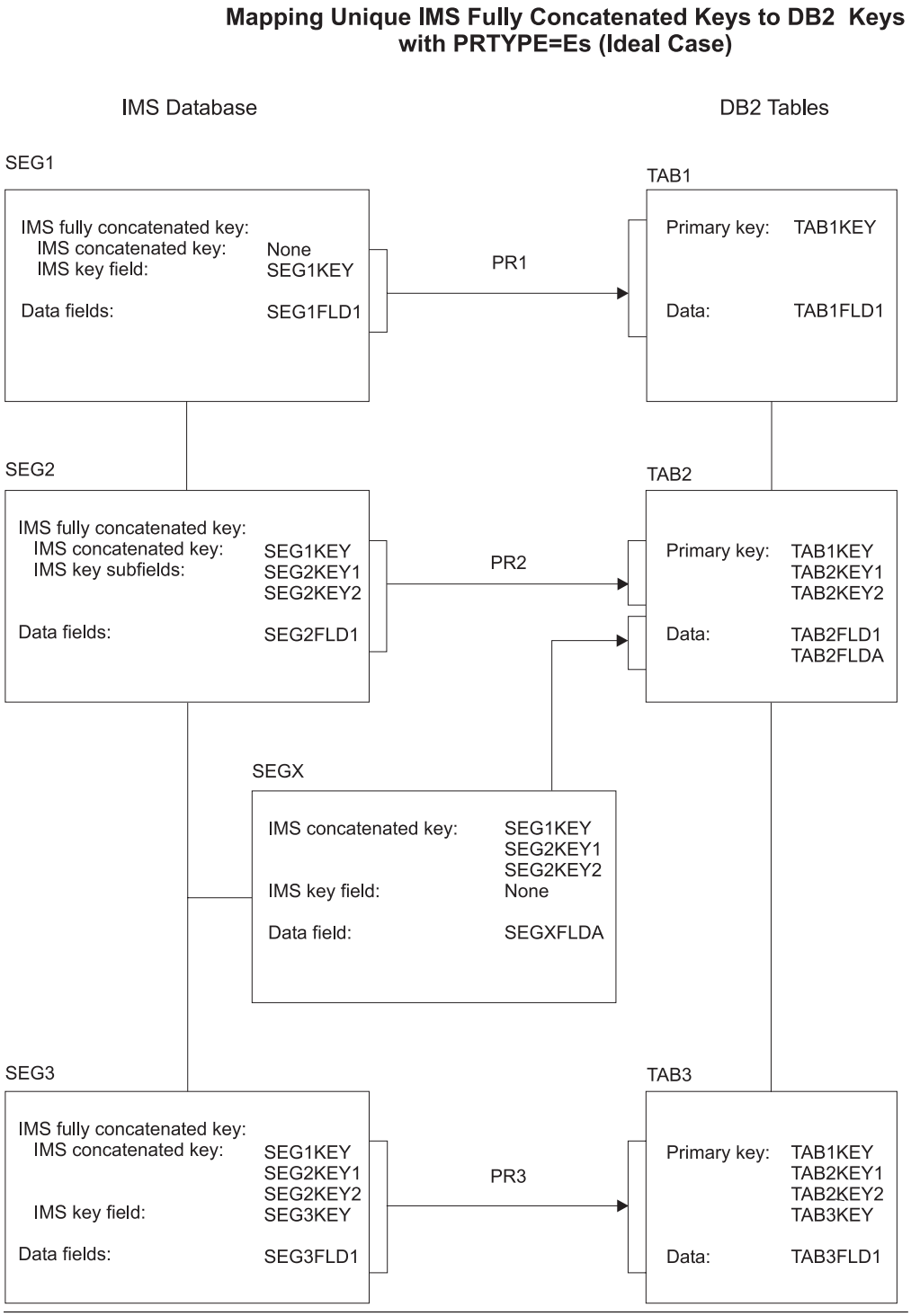


Figure 12. Mapping Unique IMS Fully Concatenated Keys to DB2 Primary Keys with PRTYPE=Es (Ideal Case)

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

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

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

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

Example of Mapping Keys in Non-Ideal Case (PRTYPE=E)

Figure 13 on page 68 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.

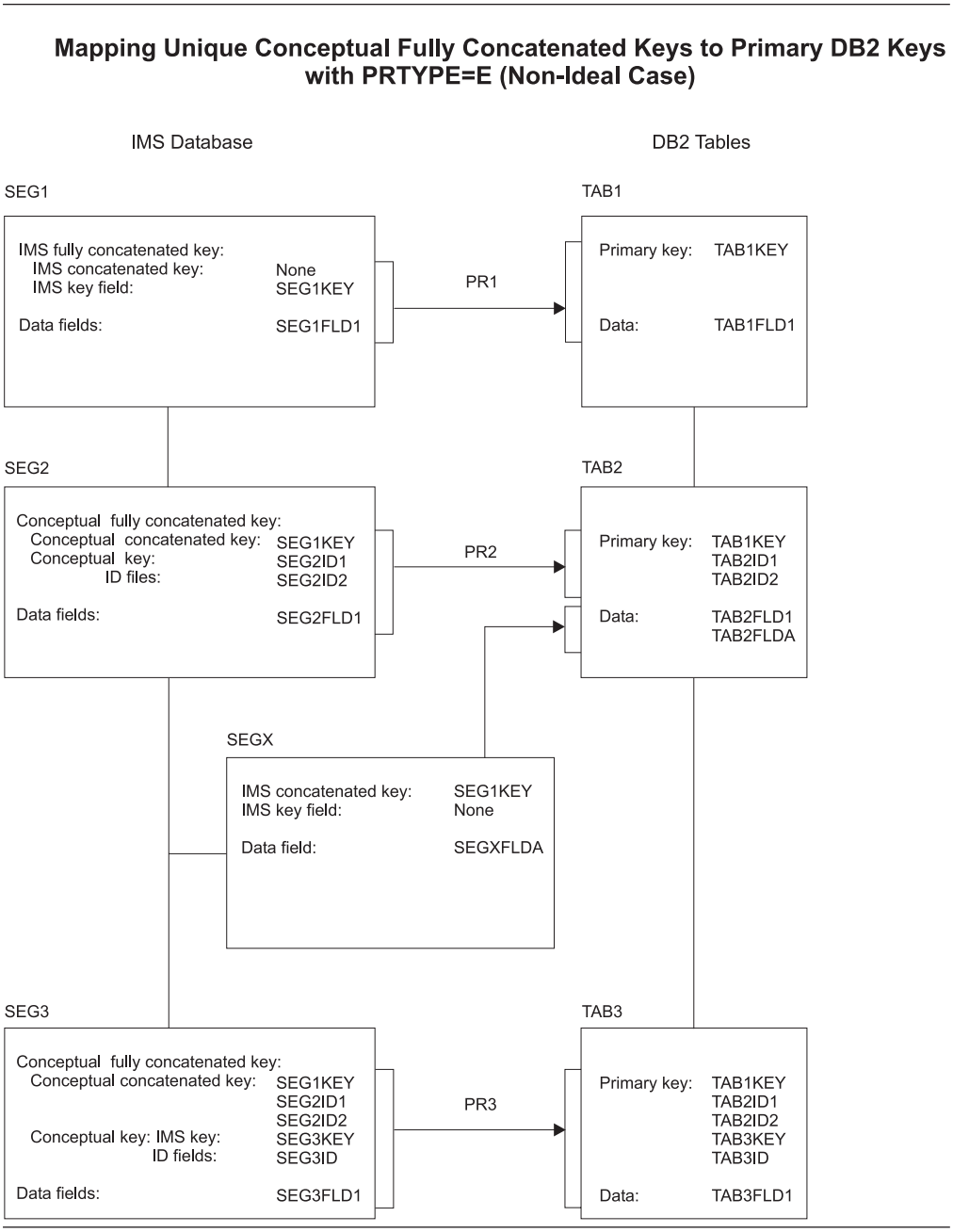


Figure 13. Mapping Unique Conceptual Fully Concatenated Keys to Primary DB2 Keys with PRTYPE=E (Non-Ideal Case)

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

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

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

- As required by key rule 3, you can identify each entity segment either through a unique IMS fully concatenated key or through a unique conceptual fully concatenated key. The DB2 primary key is either the propagated unique IMS fully concatenated key or the propagated unique conceptual fully concatenated key.
  - The entity segment SEG1 has a unique IMS key field and a unique IMS fully concatenated key: SEG1KEY. The DB2 primary key 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.
- DB2 RIRs have been implemented. As required by key rule 4, the DB2 foreign key of a child table is the propagated conceptual concatenated key of the child segment.
- According to key rule 5, CCU's direct verification cannot be used for checking the consistency of SEG2 and SEG3 with TAB2 and TAB3, because SEG2 and SEG3 have neither a unique IMS key field nor a maximum of one occurrence under their parent. You can, however, use CCU's hashing technique.

Rules For PRTYPE=L (Limited Function)

Figure 14 on page 72 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 describes how the DB2 foreign key of a dependent table should be mapped when defining DB2 RIRs that match IMS parent/child relationships.

The definition of matching DB2 RIRs is optional. If you implement RIRs, use them as a subset of IMS parent/child relationships. See “DB2 Referential Integrity Guidelines” on page 53 for more information on this subject.

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

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

– PATH data fields located in the data portion of the physical parent segment, physical ancestors

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

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

Key Rule 5: Same as PRTYPE=E and L. See page topic 64.

Key Rule 6: Same as PRTYPE=E and L. See page 65.

Example of Mapping Keys (PRTYPE=L)

Figure 14 on page 72 shows mapping keys with PRTYPE=L. In the example, the entity segment SEG2 cannot be identified uniquely by combining the IMS fully concatenated key and ID fields that do not change their value. SEG2 has, therefore, no unique conceptual fully concatenated key and cannot be propagated by using PRTYPE=E.

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

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

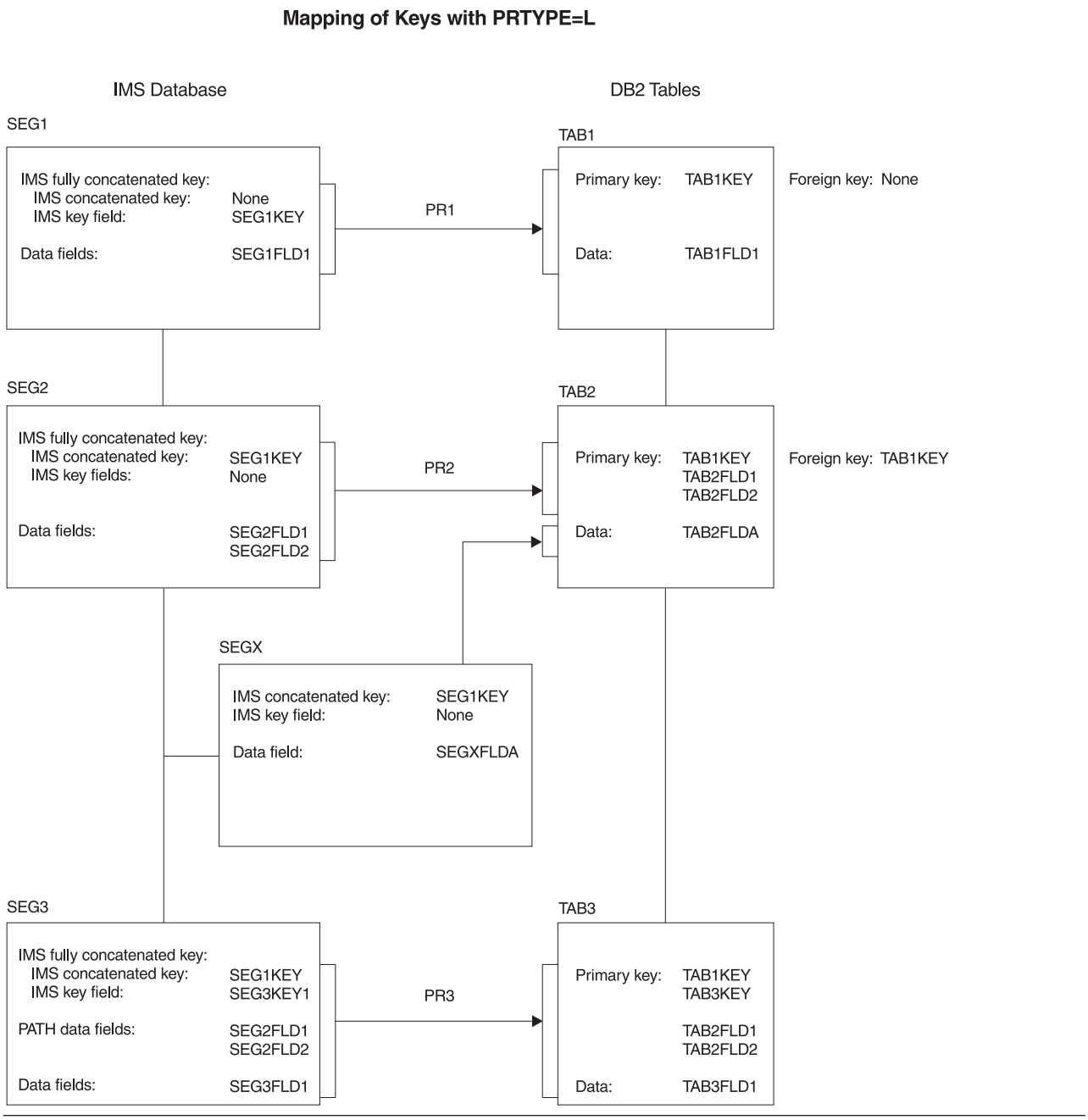


Figure 14. Mapping of Keys with PRTYPE=L

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

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

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

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

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

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

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

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

Comparison of Key Mapping Rules by Propagation Request Type

Table 5. Comparison of Key Mapping Rules by Propagation Request Type

PRTYPE=E	**PRTYPE=L**
Key Rule 1	
A propagated DB2 table must have a DB2 primary key. All columns of the DB2 primary key must be mapped by the propagation request.	Same as for PRTYPE=E.
Key Rule 2	
All of the IMS fully concatenated key of an entity segment must be propagated to the DB2 primary key. For a propagated logical child segment, the entire logical concatenated key must also be propagated.	N/A
Key Rule 3	
Ideally for each propagated entity segment: • The propagated entity segment should have a unique IMS fully concatenated key. • The DB2 primary key should be the propagated IMS fully concatenated key of the entity segment.	Same as PRTYPE=E in ideal situations.
If a propagated entity segment does not have a unique IMS fully concatenated key, then: • The entity segment must have a unique conceptual fully concatenated key.	Then: • 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

Table 5. Comparison of Key Mapping Rules by Propagation Request Type (continued)

PRTYPE=E	PRTYPE=L
• 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.
• Fields contributing to the conceptual fully concatenated key must not change their values.	• Fields being mapped to the DB2 primary key can change their value, unless the results of change violate optional DB2 RIRs or the fields are PATH data fields of a propagation request defined with PATH=ID.
Extension segments of mapping case 2 propagation requests must not have an IMS key field, participate in mapping to the DB2 primary key, nor have dependent segments propagated by PRTYPE=E.	Extension segments of mapping case 2 propagation requests must not participate in mapping to the DB2 primary key.
Key Rule 4	
The foreign key in the child table must be mapped from the same fields as the primary key of the parent table.	
The foreign key used to implement a DB2 parent/child RIR matching a *physical* IMS parent/child relationship:	
• Must be the propagated IMS/conceptual fully concatenated key of the physical parent segment.	• Is propagated from the same combination of: – Fields in the IMS fully concatenated key of the physical parent segment – PATH data fields in the data portion of the physical parent segment and/or physical ancestors as the DB2 primary key of the parent table. PATH data fields included in a foreign key should not change their value.
The foreign key used to implement a DB2 parent/child RIR matching a *logical* IMS parent/child relationship:	
• Must be the propagated IMS fully concatenated key of the logical parent segment.	• Is propagated from the same combination of fields in the IMS fully concatenated key of the logical parent segment as the DB2 primary key of the parent table.
Key Rule 5	
See description of key rule 5 on page 64.	
Key Rule 6 (A Recommendation)	
For HIDAM, HISAM, SHISAM, and HDAM and DEDBs with sequential randomizers, the DB2 index for the DB2 primary key should be clustered. The ordering sequence of this index should be the same as the IMS fully concatenated key. For HDAM and DEDBs without sequential randomizers, the DB2 table should be clustered in the same sequence as the physical HDAM/DEDB sequence, if practical.	

Supported Field Formats and Conversions

IMS DPROP supports a number of field formats and field format conversions, as shown in Table 6 on page 76. You can implement additional field formats and conversions using Field exit routines. For more information on Field exit routines, see the *IMS DataPropagator Customization Guide*.

You do not need to propagate every data field to or from the target DB2 tables. And you can map a field to multiple columns in the same table. For more information on this subject, see "Propagating a Subset of Fields or Columns" on page 47 and "Mapping Between Fields and Columns" on page 49.

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

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

Describing Fields

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

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

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

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

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

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

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

Converting Data

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

IMS DPROP supports:

- All IMS field formats supported by DXT 2.5
- All column formats supported by DB2 2.2
- Field format conversions done during an extract and load process using DataRefresher with the DB2 Load utility, except for:
  - Binary integer and decimal number to floating point
  - Floating point to binary integer and decimal number
  - Timestamp to date/time
  - Date/time fields in formats other than ISO, USA, EUR, or JIS

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

Table 6. Data Conversions Supported by IMS DPROP. "R" means a *recommended* pair of IMS field and DB2 column definitions. "S" means *supported* pair of IMS field and DB2 column definitions.

	DB2 Column Definitions													
DataRefresher and IMS DPROP Definitions of the IMS Fields	**SMALLINT**	**INTEGER**	**DECIMAL**	**FLOAT(21)**	**FLOAT(53)**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**GRAPHIC**	**VARGRAPIC**	**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 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', and X'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 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 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.
- During DB2-to-IMS propagation, an IMS field scale larger than the scale of the corresponding DB2 column can cause propagation to fail if the field is included in a WHERE clause. Do not include in the WHERE clause an IMS field with a larger scale than the corresponding DB2 column.
- 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 80 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 222.

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.
- An IMS field scale larger than the scale of the corresponding DB2 column can result in propagation failures with DB2-to-IMS propagation if the field is included in the WHERE clause. Therefore, if you are doing DB2-to-IMS propagation, do not include in the WHERE clause an IMS field with a larger scale than the corresponding DB2 column.

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'.
- For DB2-to-IMS propagation, mapping by IMS DPROP to a decimal field results in a X'C' positive and X'D' negative sign. If your installation uses signs different from X'C' and X'D' (for example, if your installation uses X'F') for a decimal IMS key field or ID field, then DB2-to-IMS propagation might fail because IMS DPROP tries to propagate to IMS segments with X'C' and X'D' signs in the key/ID. You can avoid problems by writing a Field exit routine that provides signs used in the IMS key fields and ID fields.

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.
- With mapping involving decimal fields with a fractional part, loss of the fractional part can result in problems in two-way synchronous propagation environments. You might lose fractional information in both the integer field *and* decimal field.
- Loss of the fractional part of a decimal key can cause the uniqueness of the key to be lost.
- An IMS field scale larger than the scale of the corresponding DB2 column can result if DB2-to-IMS propagation fails and the field is included in a WHERE

clause. Therefore, if you are doing DB2-to-IMS propagation, do not include in the WHERE clause an IMS field with a larger scale than the corresponding DB2 column.

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.
- DB2-to-IMS mapping by IMS DPROP to a decimal field results in a X'C' positive sign and X'D' negative sign. If your installation uses signs different from X'C' and X'D' (for example, if your installation uses X'F') for a decimal IMS key field or ID field, then DB2-to-IMS propagation might fail because IMS DPROP tries to synchronously propagate to IMS segments with X'C' and X'D' signs in the value of the key or ID fields. You can avoid problems by writing a Field exit routine that provides signs used in the IMS key fields and ID fields.

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 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 78, 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 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 32.

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

Chapter 4. Application Programs Involved in Synchronous Propagation

Synchronous propagation operates without disturbing your application programs. You are not required to make changes to application programs you use with IMS DPROP. However, this chapter presents various considerations on how your mixed-mode applications should be set to work with IMS DPROP effectively.

Topics are:
- IMS/DB2 mixed-mode processing
- IMS application checkpoint and restart
- IMS SETS with ROLS calls
- IMS logical delete rules
- IMS INIT STATUS GROUPA call
- IMS INIT STATUS GROUPB call
- SQL SET CURRENT PACKAGESET statement
- Unsupported DB2 functions in IMS/DB2 mixed-mode environment
- SQL statements in PSW key other than 8 or in authorized state

IMS/DB2 Mixed-Mode Processing

Applications involved in synchronous propagation are mixed-mode, having the ability to access both IMS and DB2 databases. Integrity between database managers is controlled by the IMS synchronization point coordinator and the two-phase commit protocol. To both database managers, an application involved in synchronous propagation appears to contain IMS and DB2 updates, even though it contains only IMS or SQL calls.

Like other mixed-mode programs, propagating applications require a DB2 plan even though they contain no SQL calls. Propagating applications also require an IMS PSB. If you are doing DB2-to-IMS propagation, the PSB must contain PCBs for the propagating IMS update calls issued by HUP.

IMS Application Checkpoint and Restart

Taking checkpoints at the proper frequency is important in batch and BMP programs. When setting your checkpoints, be aware of these considerations for mixed-mode applications:

- Concurrency needs for DB2 data.

  All DB2 data updated between two checkpoint calls is locked by DB2 and not available to other DB2 users. DB2 usually locks entire pages containing the modified rows instead of just the modified rows. Frequent checkpoint calls reduce the amount of locked DB2 data unavailable to other DB2 users.

  A non-mixed-mode IMS batch application takes checkpoints for efficient recovery and restart, however a mixed-mode batch program must be aware of the concurrency needs of other users. You might need to adjust checkpoint frequency for mixed-mode batch based on both user needs (decreasing frequency) and system performance requirements (increasing frequency).
- Time required to restart DB2 after an outage.

  After a DB2 outage (for example, after a power failure), DB2 restart first rolls back all updates of uncommitted units-of-recovery. Only after restart completes

rollback does DB2 accepts new work. If long-running batch jobs with few checkpoints were executing at the time of failure, rollback processing during DB2 restart is lengthy; and DB2 users have a long wait before DB2 accepts their work.

For example, if checkpoints for a batch job are taken every 15 minutes, rollback processing during DB2 restart may take up to 15 minutes, sometimes even longer.

The frequency with which you take checkpoints in your propagating programs can be based on:
- Expiration of a set amount of time
- Number of IMS database records or DB2 rows read
- Number of IMS database records or DB2 rows updated

Propagating IMS batch programs should always issue a final IMS symbolic checkpoint after making the last IMS/DB2 update call. This coordinates the commitment of the last UOW between IMS and DB2.

IMS batch programs involved in propagation can be restarted from only the last checkpoint they issued. When you modify a batch program or non-message-driven BMP that does not issue checkpoint calls to include checkpoint calls, remember that your modified program must be able to restart from the last checkpoint call.

For more information on program checkpoints, refer to *IMS/ESA Operations Guide*.

IMS SETS with ROLS Calls

Do not issue the IMS ROLS call with a token or the SETS call. For IMS batch regions, no form of the ROLS call should be issued. These calls are not supported in IMS/DB2 mixed-mode environments.

IMS Logical Delete Rules

Refer to "Delete Rules" on page 44 for a description of rules and the restrictions that apply to application programs.

IMS INIT STATUS GROUPA Call

An IMS application can issue INIT STATUS GROUPA calls to regain control from DL/I calls that access unavailable IMS data. INIT STATUS GROUPA calls also let the application program regain control when propagation fails.

If the application issued an INIT STATUS GROUPA call, when IMS DPROP detects a propagation failure it issues an IMS ROLB call to back out the failing UOW. Then IMS DPROP returns an error code to the application. The error code is either an IMS BB status code (IMS-to-DB2 propagation) or a -929 SQL error code and '58002' SQLSTATE (DB2-to-IMS propagation).

For additional information about the INIT STATUS GROUPA call, see "IMS INIT STATUS Call" on page 179. This section contains information about:
- ROLB calls issued by IMS DPROP
- BB status code (IMS-to-DB2 propagation)
- -929 SQL error code (DB2-to-IMS propagation)

ROLB Calls Issued by IMS DPROP

If your IMS application issues an IMS INIT STATUS GROUPA call, then it must be prepared to handle the result of the IMS ROLB calls issued by IMS DPROP when propagation fails.

The ROLB call issued by RUP and HUP:

- Backs out the failing UOW
- Resets the position of all database PCBs to the beginning of the database
- Closes all open SQL cursors, even if they have been defined as WITH HOLD

BB Status Code (IMS-to-DB2 Propagation)

If your IMS application issues an IMS INIT STATUS GROUPA call, then it must be prepared to handle a BB status code after each propagating IMS call.

When returning a BB status code, RUP also identifies the type of propagation failure by returning one of the following character strings in the segment name field of the PCB:

- PROPUNAV—unavailable resource failure
- PROPOTHR—other propagation failure. Examples of failures include non-numeric data in an IMS field defined as numeric, or an SQL not found condition when trying to replace a DB2 row.

-929 SQL Error Code (DB2-to-IMS Propagation)

If your IMS application issues an IMS INIT STATUS GROUPA call, then it must be prepared to handle the -929 SQL error code ('58002' SQLSTATE) after each propagated SQL update. When returning a -929 SQL error code, HUP also identifies the type of propagation failure by returning one of the following character strings as first token in the SQLERRM field of the SQLCA:

- PROPUNAV—indicates that a resource required for propagation was not available
- PROPOTHR—indicates some other type of propagation failure, such as numeric values outside the accepted range or a DL/I not found condition when attempting to replace an IMS segment

IMS INIT STATUS GROUPB Call

If your application issues an IMS INIT STATUS GROUPB call, it regains control from the same failures as the IMS STATUS INIT GROUPA call. Your program also regains control after the following kinds of deadlock:

- Deadlocks during nonpropagating IMS calls. This situation is described in *IMS/ESA Application Programming: DL/I Calls*.
- Deadlocks during propagating IMS calls and propagating SQL statements (only for non-message driven BMPs). The failing UOW is backed out by IMS and DB2, and your application gets control with an error code. The error code is either an IMS BC status code, if the deadlock happened while your program was issuing a propagating IMS update call, or a -911 SQL error code, '40000' SQLSTATE, if the deadlock happened while your program was issuing a propagating SQL update statement.

For additional information about the INIT STATUS GROUPB call, see"IMS INIT STATUS Call" on page 179.

SQL SET CURRENT PACKAGESET Statement

Be careful if your propagating application programs use the SQL statement SET CURRENT PACKAGESET. If the SQL statement SET CURRENT PACKAGESET is in effect during propagating updates, IMS DPROP might not be able to issue SQL statements that access the IMS DPROP directory and propagate IMS changes to DB2. And propagation fails.

Unsupported DB2 Functions in IMS/DB2 Mixed-Mode Environment

If some of your DB2-only programs update tables that are subject to DB2-to-IMS propagation, then you need to convert these programs to mixed-mode IMS/DB2 programs.

When doing the conversion, be aware that IMS/DB2 mixed-mode programs cannot use all DB2 functions available in TSO/DB2, CICS/DB2, or CAF/DB2 environments. Restrictions involve:

- SQL COMMIT and ROLLBACK statements
- DB2 functions available only with the call attachment facility (CAF)

SQL COMMIT and ROLLBACK Statements

Propagating applications should not issue SQL COMMIT or ROLLBACK statements. These statements are not supported in IMS/DB2 mixed-mode environments.

You can issue IMS checkpoint (CHKP) calls instead of SQL COMMIT statements and IMS rollback (ROLB) calls instead of SQL ROLLBACK statements.

DB2 Functions Available Only with CAF

The following DB2 functions available in a CAF environment are not available in an IMS/DB2 mixed-mode environment and cannot be used by a propagating program:

- Multi-threading of SQL statements. In an IMS/DB2 mixed-mode environment, all SQL statements must be issued from the main MVS task of your application program.
- CAF CONNECT, OPEN, CLOSE, DISCONNECT, and TRANSLATE requests. When converting to an IMS/DB2 mixed-mode environment, remove these CAF requests.

SQL Statements in PSW Key Other Than 8 or in Authorized State

Standard application programs issue SQL statements while operating in program status word (PSW) key 8 and in non-authorized MVS system state.

Some programs also issue SQL statements in authorized state or in a PSW key other than 8, which IMS DPROP does not support. The SQL communication area (SQLCA) of propagating SQL statements must be located in key 8 storage.

Chapter 5. IMS DPROP Control Information and Environment

In addition to mapping your data, as described in Chapter 2, "Decisions Affecting Mapping and Propagation," on page 11 and Chapter 3, "Propagation Guidelines, Rules, and Restrictions," on page 43, you must prepare your operating environment.

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

This chapter also provides guidance on how to prepare your environment for propagation.

IMS DPROP Control Information

This section discusses the IMS DPROP control information located in the IMS DPROP directory, propagation status file (synchronous only), and VLF objects. Topics include:
- IMS DPROP directory
- Propagation status file
- IMS DPROP's use of VLF

IMS DPROP Directory

The IMS DPROP directory consists of multiple relational tables, created during IMS DPROP generation. Figure 15 on page 91 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.

Table	Description
MSG	Can contain zero, one, or more rows. Each row contains a warning or error message issued during the creation or the latest revalidation of each propagation request.

- The following control block tables:

Table	Description
CBT	Contains one RUP Propagation Request control block (PRCB) for each propagated segment type.
HCBT	For synchronous propagation, contains one HUP Propagation Request control block (PRCB) for each: – TYPE=E defined in the IMS DPROP directory – DB2 table that is referenced by one or more PRTYPE=Us

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

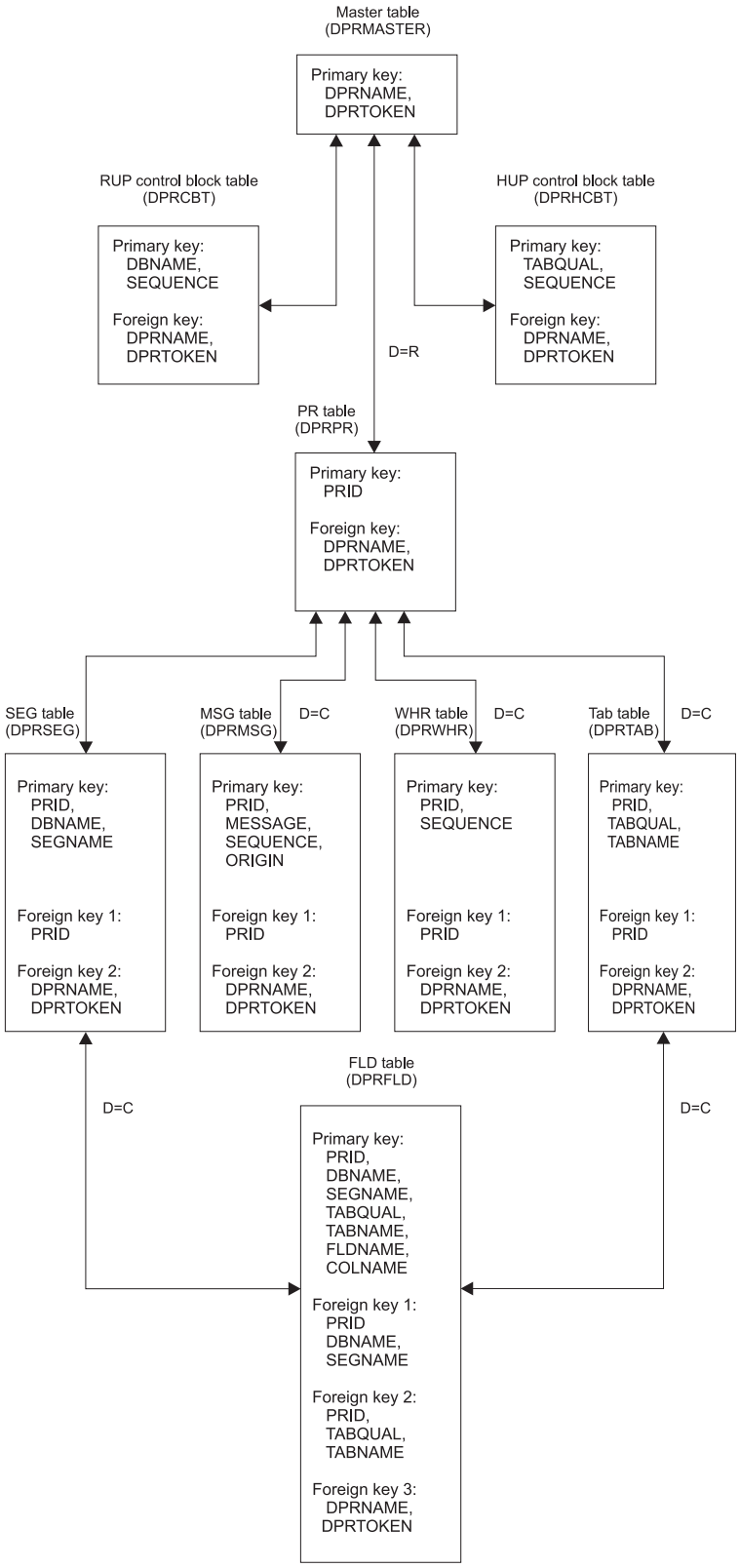


Figure 15. IMS DPROP Directory

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

With the exception of the four IMS DPROP asynchronous propagation only tables:
- RCT
- PRCT
- PRDS register table
- PRDS volume table

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

The directory tables must be updated using 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.

Propagation Status File

The propagation status file informs RUP and HUP that all IMS DPROP propagation has been stopped. You can use the status file even when DB2 or the IMS DPROP directory is not available. The status file is a sequential file with a single record containing the IMS DPROP system identifier and the IMS DPROP status indicator.

IMS DPROP's Use of VLF

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

VLF Requirements

IMS DPROP creates and uses:

- One VLF object for each propagated segment type (each RUP PRCB).
- One VLF object for each table propagated by a PRTYPE=E or U (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.
- One VLF object for the propagation status file.
- VLF objects to maintain counts of propagation failures:
  - One VLF object for each propagation request
  - One VLF object for each IMS DPROP system
- One PDS VLF object for each global master timestamp (GMTS) (in Sysplex).

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

Initializing, Refreshing or Recreating VLF Objects

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

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

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

IMS DPROP's Use of the Global Master Timestamp (GMTS) for Sysplex

IMS DPROP uses a GMTS to signal changes in control information to the various IMS DPROP components on multiple MVS (z/OS) systems. IMS DPROP implements the GMTS as a PDS (partitioned data set) member in a VLF PDS-class of objects to increase performance of its components.

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

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

For more information on Sysplex systems and how to set them up see*z/OS V1R2.0 MVS Setting Up a Sysplex* and *z/OS V1R1.0 MVS Programming Sysplex Services Reference* .

How GMTS Works

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

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

Creating and Updating the GMTS

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

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

Refreshing or Recreating the VLF PDS

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

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

JCL Changes for Sysplex IMS DPROP

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

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

Figure 16. EKYGMTS DD statement.

VLF considerations

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

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

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

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

MVG Input Tables

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

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

Audit Trail Table

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

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

IMS DPROP Operating Environment

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

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

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

In synchronous propagation, you cannot propagate a changed segment to or from tables in different DB2 systems. DB2's attachment facility supports only a single thread from an IMS dependent or batch region to a DB2 system.

IMS DPROP also restricts IMS application programs from triggering synchronous propagation activities for multiple IMS DPROP systems.

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

Multiple IMS DPROP Systems and Environments

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

Each IMS DPROP system you define has its own set of:
- Directory tables

- Propagation status file
- VLF objects and class
- Propagation requests
- SQL update modules
- DB2 plans that provide access to both the directory tables and the propagated tables

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

When defining JCL and binding DB2 plans for propagating applications or IMS DPROP utilities, be sure to provide consistent definitions. For example, all of the following control information should belong to the same IMS DPROP system:
- Propagation status file, described in the JCL
- 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 (z/OS) SVC number reserved for IMS DPROP use and MVS (z/OS) ROUTCDE used to route IMS DPROP messages to MVS (z/OS) consoles. Each IMS DPROP environment also has its own IMS DPROP load module library.

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

Scenarios for One or Multiple IMS DPROP Systems Synchronous

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

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

Scenario 1

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

Figure 17. IMS DPROP Scenario 1

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

Scenario 2

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

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

For synchronous propagation, one IMS dependent or batch region propagates using only one IMS DPROP system. As a result, an IMS region can either propagate test data or production data, not both.

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

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

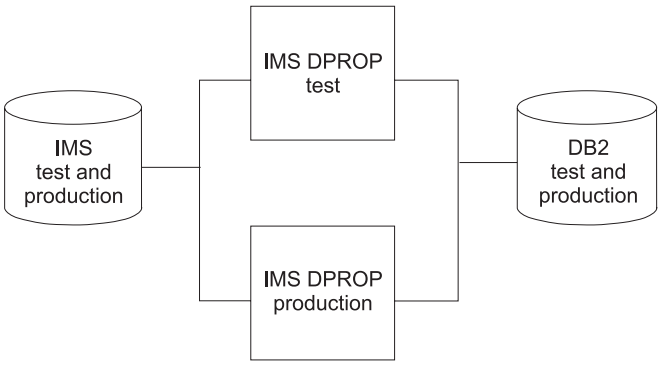


Figure 18. IMS DPROP Scenario 2

Scenario 3

The scenario in Figure 19 on page 98 shows two completely different operational environments for propagating test and production data.

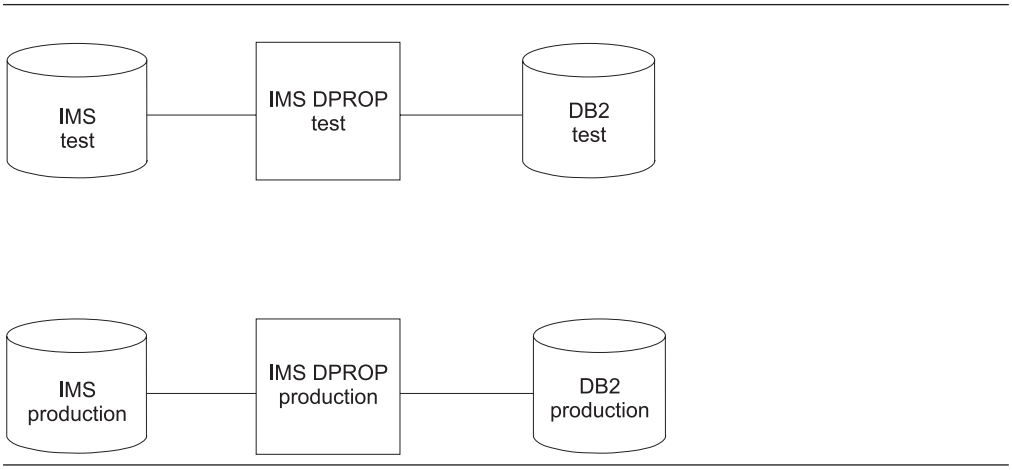


Figure 19. IMS DPROP Scenario 3

Scenario 4

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

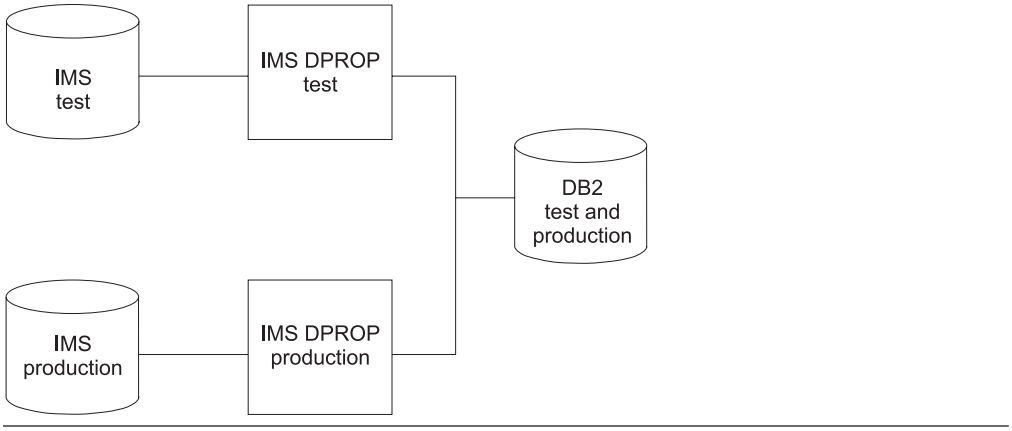


Figure 20. IMS DPROP Scenario 4

Scenario 5

The scenario in Figure 21 on page 99 shows how one IMS environment, that is used for both test and production jobs, is connected to two DB2 systems through two IMS DPROP systems.

For synchronous propagation, an IMS dependent or batch region propagates with only one IMS DPROP system. As a result, an IMS region can either propagate test data or production data, not both.

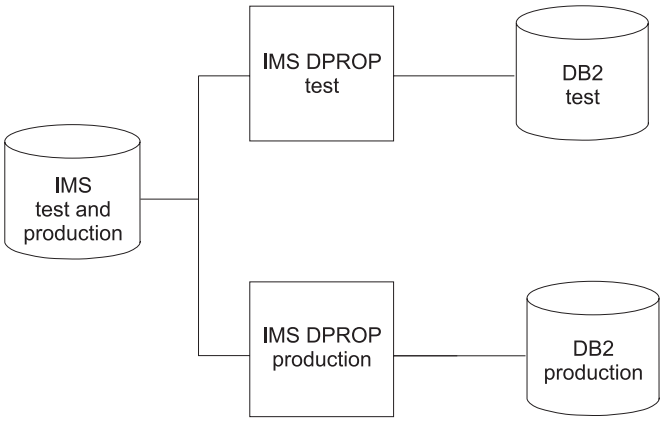


Figure 21. IMS DPROP Scenario 5

IMS Environment

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

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

Use of DBRC

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

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

- For synchronous data propagation, SCU status changes must be done in a controlled way when the affected data is not updated.
- 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.

Intersystem Data Sharing

Synchronous propagation of IMS databases involved in block-level data sharing is supported only when the updating IMS systems are on the same MVS (z/OS)

image as the DB2 system updating the propagated DB2 tables. Because DB2 support for intersystem data sharing is limited, only one DB2 system can update a shared table.

DBCTL Support of Changed Data Capture

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

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

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

Extended Recovery Facility (XRF) Considerations

Use of synchronous propagation with Extended Recovery facility (XRF) is limited because DB2 does not provide true XRF support. After an XRF takeover, DB2 must be restarted manually on the alternate XRF processor.

For synchronous IMS-to-DB2 propagation, unless propagation is emergency stopped for the whole IMS DPROP system on the alternate XRF processor, applications trying to update propagated IMS data are backed out until DB2 resources become available on the alternate XRF processor. You have two options for applications doing synchronous IMS-to-DB2 propagation:

- Accept that the propagating applications cannot be used until DB2 becomes available on the new active system
- Emergency stop propagation, start DB2, resynchronize the data, and reactivate propagation

IMS Inserts in Load Mode

When doing IMS-to-DB2 propagation for:

Synchronous propagation
IMS DPROP does not automatically propagate updates done with PCBs having PROCOPT=L or LS specified. To propagate these updates, you must provide a PROP LOAD control statement in the //EKYIN file of the IMS batch or dependent region used to load the IMS database.

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

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

Database Updates with IMS Utilities

When segments are inserted into a database using any IMS database utilities, IMS DPROP is not called. IMS DPROP does not propagate IMS segments during database reload with the IMS HD Reorganization Reload utility. Other database

utilities, including IMS HISAM Reload, Database Recovery, and DEDB Direct Reorganization utilities, also are not called.

DB2 Environment

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

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

This section discusses restrictions associated with:

- SQL updates in a non-IMS environment
- Remote SQL updates to propagated tables
- Table updates with DB2 utilities

SQL Updates in a Non-IMS Environment

IMS DPROP supports only DB2-to-IMS synchronous propagation for SQL updates done by applications running in IMS regions and using DB2's IMS attachment facility. This is a limitation of both IMS DPROP and the DB2 Data Capture function.

If some of your programs update propagated tables in non-IMS environments, consider changing their JCL so they execute in IMS regions with DB2's IMS attachment facility.

Updates to propagated tables in non-IMS environments are not propagated and cause data inconsistencies. To prevent such updates, we recommend that you set the DB2 system parameter DPROP SUPPORT to 2 as described in "Preparing DB2 for Data Propagation for DB2-to-IMS Propagation" on page 114. If you cannot set DPROP SUPPORT to 2, consider protecting your installation from inadvertent updates in non-IMS regions, as described in "Protecting Propagated Tables from Nonpropagating SQL Updates" on page 117.

Remote SQL Updates to Propagated Tables

All SQL updates to propagated tables should be issued by applications connected to the DB2 system that owns the propagated tables. Your applications should not update propagated tables owned by a remote DB2 system. "Remote" means a DB2 system other than the one connected to your application.

Remote updates to propagated tables are not propagated DB2 to IMS. Such updates cause data inconsistencies. To see how you can protect your installation from such remote updates, see "Protecting Propagated Tables from Nonpropagating SQL Updates" on page 117.

Table Updates with DB2 Utilities

Table updates done using DB2 utilities, including the DB2 Load utility, are not propagated DB2 to IMS.

CICS Environment

In a CICS environment with DBCTL, IMS DPROP supports:

- LOG-ASYNC propagation

- User asynchronous propagation with IMS Asynchronous Data Capture function only

CICS environments with local DL/I are not supported.

For synchronous propagation, IMS databases and DB2 tables that are propagated can be accessed in read-only mode by CICS applications.

Coordinating Availability of IMS Databases and DB2 Tables

For synchronous data propagation, you must carefully coordinate the availability of propagated IMS databases and DB2 tables.

With IMS-to-DB2 synchronous propagation, if a propagating IMS application is running while the IMS databases are available but the propagated DB2 tables are not, the application is backed out or abended when it tries to propagate IMS updates.

To prevent abends, you must deactivate synchronous propagation for the affected propagation requests when the DB2 resources are unavailable. You must also resynchronize the databases when the resources become available. The IMS database changes applied while synchronous propagation was deactivated are not reflected in DB2.

If the period of unavailability is short, it might be best to let the transactions abend and then reprocess them when resources are available.

Similar considerations also apply to DB2-to-IMS synchronous propagation.

Reducing Operational Risks Using ERROPT=IGNORE

You can reduce some of the risks of propagation failures affecting the availability of your applications by defining propagation requests with ERROPT=IGNORE. See "Error Handling Options" on page 177 for details. This option is usually used in only test systems.

Part 3. Setting Up for Data Propagation

Chapter 6. Setting Up Your Systems for Synchronous Propagation 105
Creating or Changing DBDs 105
EXIT Keyword (IMS-to-DB2) 106
Specifying the EXIT Keyword 106
Specifying the EXIT Keyword with Logical Child Segments. 107
Specifying Data Options on the EXIT Keyword 107
Examples of the EXIT Keyword 109
Specifying the VERSION Keyword 111
Defining the PCBs Reserved for HUP (DB2-to-IMS Synchronous Propagation) 112
Increasing CPU Time Limits of Transactions . . . 113
Converting DB2-Only Programs to Mixed-Mode IMS/DB2 Programs (DB2-to-IMS) 114
Preparing DB2 for Data Propagation for DB2-to-IMS Propagation. 114
Binding DB2 Plans: Initial Bind 115
Binding Plans with DB2 Package Bind 115
Binding Plans without DB2 Package Bind . . . 116
Creating DB2 Tables 116
Specifying Columns 116
Table Qualification. 117
Protecting Propagated Tables from Nonpropagating SQL Updates 117
One-Way IMS-to-DB2 Propagation 117
DB2-to-IMS and Two-Way Synchronous Propagation 117
Identifying to DB2 the Tables Subject to Data Capture (DB2-to-IMS Synchronous Propagation). . 118
Binding DB2 Plans for IMS-to-DB2 Synchronous Propagation: Subsequent Bind 118
Starting DB2 Monitor Trace Class 6 for DB2-to-IMS Propagation 119

Chapter 7. Defining and Changing Propagation Requests 121
Defining Propagation Requests Using DataRefresher 121
CREATE DATATYPE Command 122
CREATE DXTPSB Command 122
CREATE DXTVIEW Command 123
SUBMIT Command and EXTRACT Statement 124
DataRefresher and User Mapping Cases . . . 127
Defining Propagation Requests Using the MVG Input Tables. 128
Identifying the Propagation Request. 128
Specifying the IMS Segments to be Propagated 129
Specifying the DB2 Tables 129
Specifying the Fields 129
Executing the MVGU. 129
Propagation Parameters 131
PRTYPE—Type of Propagation Request. . . . 132
MAPCASE—Mapping Case 132
PATH—Path Data Option 132
MAPDIR—Mapping Direction. 133
TABQUAL2—DB2 Table Qualifier Used for Validation 133
ERROPT—Error Option 133
MAXERROR—Maximum Number of Reported Propagation Errors 133
ACTION 133
PRSET—Propagation Request Set Name . . . 134
PROPSUP—Propagation Suppression 134
AVU—Avoid Unnecessary Updates 134
DEFVEXT—Default Value Extension Segments: Mapping Case 2 DB2-to-IMS Only 135
KEYORDER—DB2 Key Ordering Sequence . . 135
PERFORM—Type of Operation: DataRefresher only 135
EXITNAME—Name of Propagation Exit . . . 135
PROPSEGM—Propagated Segments: User Mapping with DataRefresher Only 135
PCBLABEL—Label of IMS PCB for DB2-to-IMS Propagation Only 136
BIND—Options for a DB2 Package Bind . . . 136
Deleting a Propagation Request 136
Replacing a Propagation Request. 137
Rebuilding a Propagation Request 137
Revalidating Propagation Requests 137

Chapter 8. Granting Privileges and Authorizations for DB2 Objects 139
IMS DPROP Tables, Utilities, and Related Objects 139
Granting Privileges for IMS DPROP Tables . . 140
IMS DPROP Directory Tables 140
MVG Input Tables. 140
Audit Trail Table 140
Binding Packages of IMS DPROP Modules . . 141
Granting Privileges for IMS DPROP Collections 141
Binding Plans of IMS DPROP Utilities 142
Running IMS DPROP Utilities. 142
Additional Authorizations Required to Execute CCU 142
Additional Authorizations Required to Execute DLU 143
Additional Authorizations Required to Run MVG/MVGU 143
Additional Privileges Required to Execute the SCU 143
Additional Authorizations Required to Execute the IMS DPROP Utilities Front End Applications. 144
Propagated Tables, Propagating Applications, and Related Objects. 144
Granting Table Privileges for Propagated Tables 144
One-Way IMS-to-DB2 Propagation 144
DB2-to-IMS and Two-Way Synchronous Propagation 145
Granting Privileges for Propagating Collections 146

Binding Packages of SQL Update Modules and Propagation Exit Routines 146
Binding SQL Update Modules into Different Packages 147
Binding DB2 Plans of Propagating Applications 147
Running Propagating Applications 148
Message Processing and Fast Path Regions 148
IMS Batch and Batch Message Processing Programs 148
DB2 Sign-on Authorization Exits 148

Chapter 9. Binding and Administering Plans 149
Binding Plans with Bind Package 149
Using Different Collection IDs 150
Job Stream for Binding DB2 Packages 150
Job Stream for Binding DB2 Plans with Bind Package 152
Binding Plans without Bind Package 153
Binding Synchronous Propagation Applications 153
Initial Bind 153
Subsequent Bind 154
Binding the User Asynchronous Receiver Program 154
Job Stream for Binding DB2 Plans without Bind Package 154
DB2 ALIAS and SYNONYM Statements . . . 156
Using the CREATE ALIAS Statement . . . 157
Using the CREATE SYNONYM Statement 157
Administering DB2 Plans with or without a Resource Translation Table (RTT) 158

Chapter 10. Extracting and Loading Data for IMS-to-DB2 Propagation 159
Overview of the Extract and Load Process. . . . 159
Preventing Updates to IMS Databases 159
Using Status Change Utility (SCU) 160
Alternative to Using SCU 160
Doing the Extract and Load with DataRefresher 161
Doing the Extract and Load with Your Programs 163

Chapter 11. Extracting and Loading Data for DB2-to-IMS (DLU) Propagation 165
Overview. 165
DLU Restrictions 166
DLU Input and Output 166
DLU Input 166
DLU Output. 167
How the DLU Selects and Processes Input Data 167
Simple Scenario 169
Complex Scenarios 169
Considerations for Segments without a Unique DL/I Key 171
Considerations for Paired Segment Types 171
Physically Paired Segment Types 171
Within the Same IMS Database 171
Across Two IMS Databases 172
Virtually Paired Segment Types 172

Part 3 covers the setup phase of data propagation, including loading and extracting data. Part 3 consists of six chapters:

- Chapter 6, “Setting Up Your Systems for Synchronous Propagation,” on page 105, describes the steps involved in setting up IMS and DB2 for synchronous propagation, such as modifying JCL of propagating applications and protecting propagated tables from nonpropagated SQL updates.
- Chapter 7, “Defining and Changing Propagation Requests,” on page 121, describes how to define, change, and delete propagation requests.
- Chapter 8, “Granting Privileges and Authorizations for DB2 Objects,” on page 139, discusses granting authority for DB2 objects such as the IMS DPROP directory, MVG input tables, and DB2 plans.
- Chapter 9, “Binding and Administering Plans,” on page 149, describes binding DB2 plans and administering DB2 plans.
- Chapter 10, “Extracting and Loading Data for IMS-to-DB2 Propagation,” on page 159, explains how to extract data from an IMS database and load it into a target DB2 table.
- Chapter 11, “Extracting and Loading Data for DB2-to-IMS (DLU) Propagation,” on page 165, explains how to extract data from propagated DB2 tables and load it into an IMS database. This process can be done with the IMS DPROP DL/I Load utilities (DLU).

Chapter 6. Setting Up Your Systems for Synchronous Propagation

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

- Creating or changing DBDs
- Defining the PCBs reserved for HUP (DB2-to-IMS)
- Increasing CPU time limits
- Converting DB2 programs, if needed
- Preparing DB2 for DB2 to IMS propagation
- Binding plans, initial
- Creating DB2 tables
- Protecting tables from unwanted updates
- Identifying tables you want to propagate (DB2-to-IMS)
- Binding plans, subsequent
- Starting DB2 monitor trace class 6 for DB2 to IMS 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, see the *IMS/ESA Utilities Reference: System*. This section describes what you must do to accommodate IMS DPROP.

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

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

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

IMS-to-DB2 propagation

You must specify an EXIT= keyword to identify the propagated segments to IMS. The EXIT keyword is specified in the DBD macro or in the SEGM macros of the DBD defining the physical database. Specifying the EXIT keyword activates the IMS Data Capture function. You must specify the EXIT= keyword before propagation requests are created. See "EXIT Keyword (IMS-to-DB2)" on page 106 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 44. You might need to modify your delete rules, and make changes to the logic of your application programs.

All types of propagation

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

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

The following sections describe how you use:
- EXIT keyword
- VERSION keyword

EXIT Keyword (IMS-to-DB2)

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

On the EXIT keyword you specify whether or not:
- IMS calls one or more IMS Data Capture exit routines with the captured data.
- IMS Asynchronous Data Capture function writes captured data to the IMS log.

On the EXIT keyword, specify the exit name EKYRUP00 (IMS DPROP's RUP). 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 107.

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

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

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

Specifying the EXIT Keyword

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

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

Specifying EXIT in the SEGM macro: If only a few segments in a database are to be propagated, specify the EXIT keyword 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 keyword 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 keyword 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 synchronous propagation, IMS DPROP validates the data options for the EKYRUP00 exit routine.

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 or NOPATH and CASCADE or NOCASCADE options. IMS DPROP does not support IMS NOKEY and NODATA options.

PATH or NOPATH: The PATH or 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 or NOCASCADE: The CASCADE or 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 or NOCASCADE option.

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

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

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

PATH or NOPATH suboption of CASCADE: CASCADE includes a PATH or NOPATH suboption. The PATH or 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 or NOPATH suboption as for the PATH or NOPATH options described in "PATH or NOPATH" on page 107. 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 or NOPATH" on page 107.

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

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

Examples of the EXIT Keyword

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

In this section, examples:

1 and 2	Show how to specify EXIT at the DBD level.
5 to 8	Are similar to examples 1 to 4, except that they specify EXIT at the segment level.
9	Show how to specify a combination of synchronous and LOG-ASYNC and user asynchronous propagation of the same segment type.
10	Does not have data options explicitly specified, but default to KEY, DATA, NOPATH, (CASCADE, KEY, DATA, NOPATH). These defaults are acceptable if no propagation request is defined with PATH=ID or DENORM.
11 and 12	Specify PATH both as a data option and as a data suboption of CASCADE. Example 13 supports all propagation request definitions, including propagation requests defined as PATH=ID or DENORM.

The following examples apply to synchronous propagation.

Example 1 (synchronous propagation only)

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

The EXIT= keyword of the DBD specifies that EKYRUP00 is to be invoked. No data option is explicitly specified; the data options, therefore, default to KEY, DATA, NOPATH, (CASCADE, KEY, DATA, NOPATH). These defaults are acceptable if no propagation request is defined with PATH=ID or DENORM.

Example 2 (synchronous propagation only with path data)

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

Explicitly specifies PATH both as the data option and as the data suboption of CASCADE. The example supports all propagation request definitions, including propagation requests defined as PATH=ID or DENORM.

Example 3 (synchronous propagation only with no cascade deletes)

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

Explicitly specifies the NOCASCADE option to reduce the PATH length of data propagation. The specification of NOCASCADE was made with the assumption that IMS parent/child relationships are matched by DB2 parent/child RIRs with an ON DELETE CASCADE specification. This example defaults to the NOPATH option which is acceptable if no propagation request is defined with PATH=ID or DENORM.

Example 4 (synchronous propagation only with path data and no cascade deletes)

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

Similar to Example 3, but explicitly specifies PATH to support propagation requests defined with PATH=ID or DENORM.

Example 5 (synchronous propagation only)

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

Example 6 (synchronous propagation only)

```
SEGM NAME=segname,PARENT=psegname,BYTES=nn,EXIT=(EKYRUP00,PATH,(CASCADE,PATH))
```

Example 7 (synchronous propagation only)

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

Example 8 (synchronous propagation only)

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

Example 9 (any combination of synchronous, LOG-ASYNC and user asynchronous propagation)

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

Shows how to propagate the segment type:

- Synchronously with EKYRUP00
- User asynchronously with the LOG-ASYNC Data Capture function.
- LOG-ASYNC with no cascade deletes

Example 10 (any combination of synchronous, LOG-ASYNC, and user asynchronous propagation using ACDC)

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

Example 11 (any combination synchronous, LOG-ASYNC, and user asynchronous propagation using ACDC with path data)

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

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

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

Example 12 (any combination synchronous and user asynchronous propagation using ACDC with path data)

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

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

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

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

Specifying the VERSION Keyword

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

Depending on the DBDV keyword specified during IMS DPROP generation, the RUP or HUP (for synchronous) 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. For synchronous propagation, HUP verifies the version ID of the current DBD against the version stored in the IMS DPROP directory.

If you use the default IMS DPROP generation option, the RUP or HUP validate the full version ID. However, during IMS DPROP generation, IMS DPROP allows you to identify the portion of the version ID to use for validation. For example, you might want to use part of the ID to distinguish between DBD changes affecting propagation and those that do not, such as randomizing parameters.

The following examples illustrate how you can use the VERSION keyword.

- When an IMS DPROP-related DBD change is made, you can alter the part of the version ID that was defined to IMS DPROP for version checking. After performing the DBDGEN, you must:
  1. Modify the propagation request definitions that are affected by the change
  2. Regenerate all propagation requests for the altered DBD
  3. If the DBD change has an effect on mapping or propagation, re-extract the affected data from the:
     - IMS database (IMS-to-DB2 propagation only)
     - DB2 tables (DB2-to-IMS propagation only)
- If you make a change to the DBD that does not affect propagation or mapping, the IMS DPROP-specific part of the version ID should not change.
- If the DBD change includes a change to the IMS DPROP-related part of the version ID but has no impact on mapping or propagation, propagation requests must be regenerated; however, you do not need to re-extract the data. You can use the PERFORM=BUILDONLY propagation parameter when you regenerate the propagation request with DataRefresher.
- If you are using VLF, you must enter the SCU control statement INIT VLF as described in the *IMS DataPropagator Reference*.

For more information on the VERSION keyword of the DBD macro, refer to *IMS/ESA Utilities Reference: System* and *IMS DataPropagator Reference*.

Defining the PCBs Reserved for HUP (DB2-to-IMS Synchronous Propagation)

For DB2-to-IMS synchronous propagation, you must define HUP PCBs exclusively for synchronous propagation. The HUP uses *HUP PCB*s, defined in the IMS PSB of each propagating program, to issue the IMS calls that propagate DB2 changes to IMS. Your Propagation exit routines can also use HUP PCBs when you are doing user mapping.

You must define in the PSB at least one HUP PCB for each physical IMS database to which data is to be propagated. Rules for defining HUP PCBs are:

1. HUP PCBs must be created with a 1- to 8-character PCB name. The PCB name is specified during PSBGEN either as an Assembler label on the PCB macro or in the PCBNAME= keyword of the PCB macro.

   You should establish naming conventions for HUP PCBs to fulfill the following IMS and IMS DPROP requirements:

   - IMS requires that PCB names be unique within a PSB.
   - IMS DPROP requires that the PCB name be the same as the value in the PCBLABEL propagation parameter of the propagation request definition. The names of HUP PCBs used to propagate a specific propagation request must be identical in the PSBs of propagating programs.

   Examples of naming conventions for HUP PCBs are:

   - Use the name of the physical DBD if the DBD names are not already used as PCB names in programs that do DB2-to-IMS synchronous propagation.
   - Use a special combination of characters for the first part of the HUP PCB name, for example, HUP or EKY. For the second part of the name use either an abbreviation of the database name or some number identifying the propagated database.
2. HUP PCBs should refer to the physical DBD.
3. HUP PCBs should access the physical database through its primary processing sequence. Therefore, do not specify a PROCSEQ= keyword on the PCB macro or an INDICES= keyword on SENSEG macros.
4. HUP PCBs should be sensitive to all segments to which data is be propagated and to their physical parents and ancestors. Therefore, include in the PCB one SENSEG statement for each propagated segment and its physical parents and ancestors.
5. HUP PCBs should specify the processing option PROCOPT=A.
6. HUP PCBs should not use field level sensitivity.
7. Specify LIST=NO on the PSBGEN PCB macro to avoid changes to the PCB list passed by IMS on entry to the application program. Changes to the PCB list causes changes to your application programs.

If you are running your propagating program in online regions, you must define the PSB to the IMS online system and do an ACBGEN.

To include HUP PCBs in the PSB of your propagating programs you could:

1. Define for each propagated physical database a *model HUP PCB* that is sensitive to all propagated segments in the database or all segments in the database.

2. Store the PCB source definition as a member in a partitioned data set.
3. Include in the PSB source of each propagating program the source of the model HUP PCB using, for example, an Assembler COPY statement.

Figure 22 shows how to add HUP PCBs to an existing PSB.

```
         PCB    TYPE=...
         PCB    TYPE=...
         SENSEG NAME=...

 *        THE FOLLOWING PCBs ARE ADDED FOR HUP

         PCB    TYPE=DB,PCBNAME=hupdb1,LIST=NO,                    *
                DBDNAME=imsdb1,KEYLEN=length,PROCOPT=A
         SENSEG NAME=seg1a,PARENT=0
         SENSEG NAME=seg1b,PARENT=seg1a
         SENSEG NAME=seg1c1,PARENT=seg1b

hupdb2   PCB    TYPE=DB,LIST=NO,                                   *
                DBDNAME=imsdb2,KEYLEN=length,PROCOPT=A
         SENSEG NAME=seg2a,PARENT=0
         SENSEG NAME=seg2b1,PARENT=seg2a
         SENSEG NAME=seg2b2,PARENT=seg2a
         SENSEG NAME=seg2b3,PARENT=seg2a

*        END OF HUP PCBs

         PSB    LANG=....,PSBNAME=....
         END
/*
```

Figure 22. Adding HUP PCBs to an Existing PSB

Increasing CPU Time Limits of Transactions

You may need to increase the CPU time limit for IMS transactions involved in propagation; additional time is required to capture the changed data.

You also might need to increase the CPU limit because:

- If doing IMS-to-DB2 synchronous propagation, time is required to process the SQL statements that propagate the IMS changes.
- If doing DB2-to-IMS synchronous propagation, time is required to process the IMS calls that propagate the DB2 changes. And when the first propagating transaction is executed in a given message region, more CPU time is required to initialize IMS DPROP.

The CPU time limit of each transaction code is specified on the PROCLIM= keyword of the TRANSACT macro at IMS system definition. PROCLIM can also be altered by IMS operator commands. For more information on the TRANSACT macro, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Converting DB2-Only Programs to Mixed-Mode IMS/DB2 Programs (DB2-to-IMS)

For DB2-to-IMS synchronous propagation, if some of your DB2-only programs update propagated tables, you must convert the programs to mixed-mode IMS/DB2 programs.

When doing the conversion:

- Check that the rules described in Chapter 4, "Application Programs Involved in Synchronous Propagation," on page 85 are observed. If necessary, modify your programs.
- Link edit these programs with the DB2 language interface used with DB2's IMS attachment facility. Do not use the DB2 language interface used with DB2's TSO or CICS attachment facility or the call attachment facility (CAF).
- Create an IMS PSB for your mixed-mode IMS/DB2 program. The PSB must contain a PCB for each physical IMS database to which data is propagated. The IMS PCBs are reserved for the HUP updates to the IMS databases. For more detail on these PCBs, see "Defining the PCBs Reserved for HUP (DB2-to-IMS Synchronous Propagation)" on page 112.
- If your program runs in an online region, define the IMS PSB to the IMS online system. And perform an ACBGEN.
- Convert the current JCL of your DB2-only application into JCL for IMS batch, BMP, MPP, or IFP mixed-mode propagating regions. Refer to the following documentation for the JCL requirements for these kinds of regions:
  - For the IMS requirements: *IMS/ESA Installation Volume 2: System Definition and Tailoring*
  - For the DB2 requirements: "JCL Changes for DB2" on page 245
  - For the IMS DPROP requirements: "JCL Changes for Synchronous Propagation" on page 243

Preparing DB2 for Data Propagation for DB2-to-IMS Propagation

You prepare DB2 for synchronous propagation by setting the DB2 system parameter DPROP SUPPORT to 2 or 3. This parameter is set on the DB2 installation panel either at or after DB2 installation time. See the *DB2 Administration Guide* to understand how this parameter is set.

For data integrity reasons, we strongly recommend you set the IMS DPROP support parameter to 2, not 3 because:

- Setting the DPROP SUPPORT parameter to 2 protects your propagated tables from being updated but not propagated. Setting the parameter to 2 prevents updates to DB2 tables created with DATA CAPTURE CHANGES option when either:
  - The update is done in a non-IMS environment.
  - Monitor trace class 6 is stopped.

  When the DB2 system parameter DPROP SUPPORT is set to 2, *all* updates to the DB2 tables must be made in an IMS environment through either a user-written program or the IMS DPROP-provided EKYTIAD program.

  EKYTIAD ensures that the updates are made in an IMS environment by acting as an IMS application. EKYTIAD calls DSNTIAD, so that SQL statements supplied by you or by the CCU can be processed.

- Setting the DPROP SUPPORT parameter to 3 can cause accidental updates to propagated tables; the changes are propagated. This can happen:
  - When propagated tables are updated in a non-IMS environment, for example, in a TSO environment with QMF
  - If monitor trace class 6 has been unintentionally stopped, for example, when a DB2 operator enters a STOP TRACE command with the wrong class number

  If you use software requiring that DPROP SUPPORT be set to 3, consider protecting your propagated data as described in "Protecting Propagated Tables from Nonpropagating SQL Updates" on page 117.

Even if DPROP SUPPORT is set to 2 or 3, you can still stop some or all synchronous propagation if necessary by using SCU ESTOP, DEACTIVATE, or EDEACTIVATE control statements.

Binding DB2 Plans: Initial Bind

For synchronous propagation, you need to bind the DB2 plans of your propagating application programs. The DB2 plans must at least provide access to the IMS DPROP directory tables for RUP and HUP.

You must bind the plans *before* you:

- Do the DBDGEN changes that activate IMS data capture of database changes (IMS-to-DB2 propagation)
- Provide the CREATE TABLE or ALTER TABLE specifications on the DATA CAPTURE CHANGES option that request DB2 capture of table changes (DB2-to-IMS propagation)

After making the changes to DBDGEN and the table specifications, the RUP or HUP (for synchronous) is called when affected IMS databases and DB2 tables are updated, even though you might not have defined any propagation requests yet. When called, the RUP or HUP must have access to the IMS DPROP directory. Therefore, any application that updates the affected databases or tables must have an application plan bound for IMS DPROP directory access before you make the changes to DBDGEN or table specifications. Referred to as the *initial application bind*, this bind must be done or updating applications fail.

The bind of the application program plan can be done with or without use of the package bind function.

Binding Plans with DB2 Package Bind

Using the DB2 package bind function makes administration of your DB2 plans easier. If you use package bind, you do not to re-bind the DB2 plans of your propagating applications after creation of or changes to propagation requests and your exit routines.

To use package bind, you must bind the following DBRMs as DB2 packages:

- DBRMs of IMS DPROP modules accessing the IMS DPROP directory. This package bind is done as part of the generation and installation of IMS DPROP.
- DBRMs of the IMS DPROP SQL update modules. One SQL update module exists for each propagation request in a generalized mapping case. Each SQL update module has a DBRM. The package bind of the DBRMs is done later

when you create the propagation requests; the package bind can be automatically done by MVG as part of propagation request creation.
- DBRMs of your IMS DPROP exit routines, for example Propagation exit routines, that issue SQL update statements. Usually you bind the package of the DBRMs after precompiling and compiling of your exit routines.

When binding the plan of your propagating applications, you list in the PKLIST keyword of the BIND command the names of the package collections containing the preceding DB2 packages.

Details about binding propagating applications are discussed in Chapter 9, "Binding and Administering Plans," on page 149.

Binding Plans without DB2 Package Bind

To bind the plans of your propagating applications, you list in the MEMBER keyword of the BIND command the names of the RUP and HUP DBRMs that access the IMS DPROP directory.

Later, if you make changes you must re-bind the plans of your propagating applications. After creating or changing your propagation requests and exit routines that issue SQL statements, you must include in the plans the DBRMs of:
- IMS DPROP SQL update modules
- Your IMS DPROP exit routines, for example Propagation exit routines, that issue SQL update statements

If no updates are made until after propagation requests are defined, then the bind is optional.

Details about binding propagating applications are discussed in Chapter 9, "Binding and Administering Plans," on page 149.

Creating DB2 Tables

DB2 target tables (model target tables) must exist before you create the IMS DPROP propagation requests. For local MVS image, target tables must be created on the target MVS image. For remote MVS image, target tables must be created on the target MVS image.

To create your propagated DB2 tables and establish RIRs among them, you need to code and execute the appropriate SQL statements. For more information on the SQL CREATE TABLE statement, refer to *DB2 Administration Guide and DB2 SQL Reference*.

For IMS DPROP, you must:
- Define columns
- Determine if you are to use qualified or unqualified tables

Specifying Columns

Define primary key columns as NOT NULL or NOT NULL WITH DEFAULT. Columns that are not mapped from IMS, or that are mapped from IMS fields that may validly be absent at propagation time, should be defined to permit null values or as NOT NULL WITH DEFAULT. You can use null or NOT NULL WITH DEFAULT for:
- Fields of IMS extension segments using mapping case 2
- Fields of variable-length segments

- Data mapped with a user mapping case

Table Qualification

If you are creating propagation requests with *qualified* table names, define the tables before you create propagation requests. If you are creating propagation requests with *unqualified* table names, then you must create *model* tables before you create propagation request.

Protecting Propagated Tables from Nonpropagating SQL Updates

Nonpropagating SQL updates to propagated tables can cause inconsistencies between the IMS and DB2 copy of propagated data. Develop a protective strategy to prevent unintentional nonpropagating SQL updates. Distinguish between:

- One-way IMS-to-DB2 propagation
- DB2-to-IMS and two-way synchronous propagation

One-Way IMS-to-DB2 Propagation

When implementing one-way IMS-to-DB2 propagation, SQL updates are *not* propagated to IMS and usually result in inconsistencies between IMS and DB2. To protect your propagated tables against such updates:

- Restrict table privileges beyond SELECT to the few individuals who need the privileges (see "Granting Table Privileges for Propagated Tables" on page 144 for details).
- Restrict execution of DB2 plans of propagating MPPs, message-driven BMPs, and IFPs to IMS online systems (see "Message Processing and Fast Path Regions" on page 148 for details).
- Restrict execution of DB2 plans of propagating BMP and batch programs to a few user IDs or a few functional identifiers.

DB2-to-IMS and Two-Way Synchronous Propagation

When implementing DB2-to-IMS or two-way synchronous propagation, the following SQL updates to tables marked for data capture are *not* propagated:

- SQL updates issued by programs or tools that do not use an IMS connection to DB2
- Remote SQL updates issued from a Distributed Data Facility (DDF) connection
- SQL updates issued when tracing for monitor class 6 is not active, for example, after a -STOP TRACE(MONITOR) CLASS(6)

Such SQL updates usually result in inconsistencies and should be avoided by setting the DB2 DPROP SUPPORT system parameter to 2. See "Preparing DB2 for Data Propagation for DB2-to-IMS Propagation" on page 114 for further discussion. If you cannot set the DPROP SUPPORT parameter to 2 because of your site requirements, consider:

- Defining a DB2 Validation exit routine for the propagated tables. Use the VALIDPROC clause of the CREATE TABLE or ALTER TABLE statement.

  Your DB2 Validation exit routine gets a description of the type of DB2 connection from the exit-specific parameter list described by the DB2 macro DSNDRVAL. The type of DB2 connection is in the RVALCSTC field.

  The Validation exit routine can reject updates issued by non-IMS connections.

  For more information on DB2 Validation exit routines, refer to DB2 Administration Guide.

- Restrict TRACE, SYSADM, and SYSOPR authorities to a few IDs to limit the number of IDs that can issue the DB2 command *-STOP TRACE(MONITOR) CLASS(6)*.

Identifying to DB2 the Tables Subject to Data Capture (DB2-to-IMS Synchronous Propagation)

For DB2-to-IMS synchronous propagation, you need to identify to DB2 the tables you want to propagate by using the DB2 DATA CAPTURE CHANGES option on the CREATE TABLE or ALTER TABLE statement.

Ensure you start the DB2 trace for monitoring class 6. For details, refer to "Starting DB2 Monitor Trace Class 6 for DB2-to-IMS Propagation" on page 119.

Binding DB2 Plans for IMS-to-DB2 Synchronous Propagation: Subsequent Bind

For IMS-to-DB2 synchronous propagation, use the DB2 package bind function so that you don't have to re-bind the DB2 plans of your propagating applications after creating or changing your propagation requests and IMS DPROP exit routines.

As described in Chapter 9, "Binding and Administering Plans," on page 149, all applications involved in synchronous propagation must have an application plan that is bound so RUP has access to the IMS DPROP directory tables. Once a propagation request has been defined for a database, one or more additional DBRMs must be included in the bind of the plan for application programs that modify that database.

If you are defining a propagation request for a generalized mapping case, IMS DPROP generates an SQL update module. IMS DPROP calls the DB2 pre-compiler to process the SQL update module, and then assembles and links the update module. The output of the DB2 pre-compiler is a DBRM that must be bound with the propagating application's plan.

If you have written any IMS DPROP exit routines that issue SQL statements, the exit routines must also be pre-compiled and their DBRMs bound with the application plan. For example, if you have written a Propagation exit routine, you issue updating SQL calls from the exit. Therefore, you must pre-compile your Propagation exit routine and bind the application plan with the DBRM of the Propagation exit routine.

If you do not use the DB2 package bind function, you must re-bind the application plans using the propagation request whenever a propagation request is defined or changed. You must re-bind the plans with the DBRM of the new SQL update module produced by IMS DPROP before the propagation request is activated.

Details about binding propagating applications are discussed in "Binding DB2 Plans: Initial Bind" on page 115.

Starting DB2 Monitor Trace Class 6 for DB2-to-IMS Propagation

To enable DB2-to-IMS synchronous propagation, you also must start monitor tracing for class 6 by doing one of the following tasks:

- At DB2 installation time, set the appropriate installation parameter
- Use the DB2 -START TRACE(MONITOR) CLASS(6) command
- Issue a DB2 IFI COMMAND request from an application program

In an earlier step (see page "Identifying to DB2 the Tables Subject to Data Capture (DB2-to-IMS Synchronous Propagation)" on page 118), you identified to DB2 which tables are subject to DB2 Data Capture by specifying DATA CAPTURE CHANGES on the CREATE TABLE or ALTER TABLE statement. Until the step is done, no DB2 changes are captured and the IMS DPROP-provided DB2 Data Capture exit routine (HUP) is not invoked. Therefore, you modify the JCL and plans of your DB2 applications to meet HUP and IMS DPROP requirements later in the setup for propagation process.

In contrast to the IMS Data Capture function where multiple exits can be specified on the DBDGEN, DB2 supports only one DB2 Data Capture exit routine for each IMS region. If you are using IMS DPROP for DB2-to-IMS synchronous propagation, you use the IMS DPROP-provided HUP, and name it DB2CDCEX (the alias of HUP). If your installation needs to have HUP coexist with another generalized DB2 Data Capture exit routine, you might want to implement the other generalized routine as an IMS DPROP DB2 Data Capture subexit routine.

Chapter 7. Defining and Changing Propagation Requests

Creating a propagation request involves specifying the IMS fields (or data elements), IMS segments, DB2 columns, and DB2 tables to be propagated. The information specified associates keys and data from IMS databases to DB2 tables. You can code propagation requests either by using both DataRefresher and IMS DPROP and defining propagation requests with DataRefresher, or by defining propagation requests in the MVG input tables.

To define a propagation request in the MVG input tables, you can provide an application that extracts the necessary information from a dictionary system, or you can use QMF or SPUFI. If you use DataRefresher to build your propagation requests, you do not need to create MVG input tables.

This chapter covers:
- Defining propagation requests using DataRefresher
- Defining propagation requests using the MVG Input Tables
- Propagation parameters
- Deleting a propagation request
- Replacing a propagation request
- Rebuilding a propagation request
- Revalidating propagation requests

Defining Propagation Requests Using DataRefresher

Use both IMS DPROP and DataRefresher to:
- Define IMS DPROP propagation requests using DataRefresher User Input Manager (UIM) commands
- Do the IMS extract and DB2 load process with the DataRefresher Data Extract Manager (DEM) and the DB2 LOAD utility

See the DataRefresher library for details of the DataRefresher UIM and DEM.

This section describes the definition of propagation requests with DataRefresher UIM; see Chapter 10, "Extracting and Loading Data for IMS-to-DB2 Propagation," on page 159, for a description of the extract and load with DataRefresher DEM.

By combining IMS DPROP and DataRefresher, you can use the same mapping definitions for data propagation and extract and load. The mapping and data conversions done by IMS DPROP for generalized mapping cases during propagation are a compatible subset of the mapping and conversions done by DataRefresher during the extract and load. Compatibility is important to avoid propagation failure and data inconsistency.

You can also use DataRefresher to provide mapping definitions for one-way DB2-to-IMS synchronous propagation, or even if you do not intend to extract IMS data with DataRefresher.

When DataRefresher has an extract request for which data propagation is to be done, the IMS DPROP Map Capture exit (MCE) validates the extract request, which is then called a propagation request.

Before defining propagation requests with DataRefresher, you need to describe the IMS data to be extracted and propagated to DataRefresher. Use the following DataRefresher UIM commands:

- CREATE DATATYPE commands, if you need to use Field exit routines. See "CREATE DATATYPE Command" on page 122.
- 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 122.
- 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 123.

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 124. 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 23 on page 127 illustrates the process. For complete information on using DataRefresher commands to define propagation requests, refer to the *IMS DataPropagator Reference*. This section provides more information on:

- The CREATE DATATYPE command
- The CREATE DXTPSB Command
- The CREATE DXTVIEW command
- The SUBMIT command and EXTRACT statement
- DataRefresher and user mapping cases

CREATE DATATYPE Command

CREATE DATATYPE commands are optional. Use them only if you intend to use Field exit routines because:

- Your IMS database contains fields in formats not supported directly by IMS DPROP and DataRefresher
- You want to perform data conversions that are not directly supported by IMS DPROP and DataRefresher

Each CREATE DATATYPE command defines a unique two-character name that identifies a user data type and associates a Field exit routine with it.

The DataRefresher UIM records your CREATE DATATYPE definitions in the DataRefresher FDTLIB data set.

CREATE DXTPSB Command

The CREATE DXTPSB command describes your IMS databases, segments, and fields to DataRefresher. Usually, you describe each IMS database (or each group of IMS databases if the DXTPSB contains multiple DXTPCB statements) to DataRefresher only once.

The CREATE DXTPSB command includes:

- One or more DXTPCB statements. The DXTPCB statement names the physical IMS database to be extracted and propagated.

- One or more SEGMENT statements. The SEGMENT statement names the physical segment to be extracted and propagated, as well as its physical parent and ancestors. The statement also indicates whether a Segment exit routine needs to be called.

  If you are using mapping case 3 propagation requests to propagate segments containing embedded structures, you also use one SEGMENT statement to describe each embedded structure. Embedded structures are called *internal segments* in this book.
- Multiple FIELD statements. The FIELD statement assigns a symbolic name to each field and describes the field in detail. For example, the statement describes the data format, length, and starting position of the field within the segment.

  If a segment is not processed by an IMS DPROP Segment exit routine, the definitions you provide in the FIELD statement should describe the fields of the segment as they appear in the I/O area of an IMS call.

  If a segment is processed by an IMS DPROP Segment exit routine, then the definitions you provide on the FIELD statements describe the fields in the edited format of the segment. The edited format is:
  - For IMS-to-DB2 mapping, the segment format *after* editing by the Segment exit routine
  - For DB2-to-IMS mapping, the segment format *before* 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. The *IMS DataPropagator Reference* describes in detail which options are supported by IMS DPROP.

CREATE DXTVIEW Command

Each CREATE DXTVIEW command describes one hierarchical path of the database from which IMS data is extracted and propagated. You can also use DXTVIEW commands to identify a subset of the IMS fields described in the CREATE DXTPSB.

Each CREATE DXTVIEW refers to a DXTPCB. On the CREATE DXTVIEW command, you identify which fields of one hierarchical path should be included in the view.

As described in the *IMS DataPropagator Reference*, you need to provide at least one CREATE DXTVIEW command for each hierarchical path of the IMS database

containing segments to be extracted and propagated. For propagation requests belonging to mapping case 2, you need to provide one CREATE DXTVIEW command for each extension segment type. The DataRefresher UIM stores the definitions you provide in CREATE DXTVIEW commands in the FDTLIB data set for later reference.

SUBMIT Command and EXTRACT Statement

After creating the DATATYPEs, DXTPSBs, and DXTVIEWs, you can define propagation requests by providing one DataRefresher SUBMIT command with a DataRefresher EXTRACT statement for each propagation request to be defined. In DataRefresher, the propagation requests being defined are called extract requests.

The SUBMIT command assigns an eight-byte propagation request identifier (PR ID) to the propagation requests. The PR ID is also used as the name of the SQL update module that IMS DPROP generates whenever the propagation request, defined with MAPDIR=HR or TW, uses one of the generalized mapping cases.

The DataRefresher EXTRACT statement:

- Identifies the name of the DB2 table
- Refers to one or more DXTVIEWs
- Associates each IMS field to be propagated with a DB2 column
- Refers to only one DXTVIEW, for mapping case 1
- Refers to one DXTVIEW for each extension segment, for mapping case 2

To define propagation requests, you must specify a MAPEXIT=EKYMCE00 keyword on the DataRefresher SUBMIT command. The DataRefresher UIM then calls the IMS DPROP-provided Map Capture exit routine EKYMCE00. You should also provide IMS DPROP-specific information—such as the mapping case number and, for user mapping cases, the name of a Propagation exit routine—either on the MAPUPARM keyword of the SUBMIT statement or in a data set containing IMS DPROP default values. IMS DPROP-specific information is described in “Propagation Parameters” on page 131.

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. For DB2-to-IMS synchronous propagation, IMS DPROP calls the Propagation exit routine associated with the propagation request each time the table identified on the DataRefresher EXTRACT statement 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, "IMS DPROP 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.
- For synchronous propagation, one HUP propagation request control block (PRCB) is created for each DB2 table propagated by a PRTYPE=E or PRTYPE=U specifying MAPDIR=RH or TW. The control block contains mapping information for all propagation requests propagating from or to a table. 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, HUP PRCBs are located in both the IMS DPROP directory and in 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 RUP and HUP. If RUP or HUP detect changes to the directory, they refresh directory objects stored in memory so the changes become effective. The unique row of the master table is also stored in VLF to improve performance.
- For propagation requests belonging to a generalized mapping case and specifying MAPDIR=HR or TW, MVG also generates the assembler source code for an SQL update module. The module contains all the SQL update statements required to propagate data from IMS to DB2 based on the propagation request definitions. The SQL source is pre-compiled, assembled, and linked into a load library as an SQL update module by IMS DPROP. You must then use a DB2 BIND operation to bind the DBRM of the SQL update module into either a DB2 package or the plans of propagating applications. You must do the bind before the DBRM can be used in propagation. You might want to bind the DB2 package automatically by using MVG.

If the propagation request is created without errors, IMS DPROP returns to the calling DataRefresher UIM. UIM then stores the corresponding extract request in the DataRefresher EXTLIB data set. The definitions are then available in EXTLIB to the DataRefresher DEM when an extract is done.

You should save your DataRefresher SUBMIT and EXTRACT specifications, because you need to provide them to DataRefresher every time you want to extract IMS data with DataRefresher. After successful completion of the extract, the DEM

deletes the extract request from EXTLIB. However, IMS DPROP keeps the propagation request definitions in the IMS DPROP directory until you delete them using the IMS DPROP MVGU.

You can use DataRefresher to build the propagation request, without extracting IMS data with the DataRefresher DEM. To do so, specify PERFORM(BUILDONLY) in the MAPUPARM keyword of the DataRefresher EXTRACT statement.

If you use the DataRefresher UIM to define propagation requests, both IMS DPROP and DB2 functions are called. UIM JCL needs to be modified with JCL required by IMS DPROP and DB2. The propagation request definition process using DataRefresher is illustrated in Figure 23 on page 127.

Refer to the *IMS DataPropagator Reference* for more detailed information and examples.

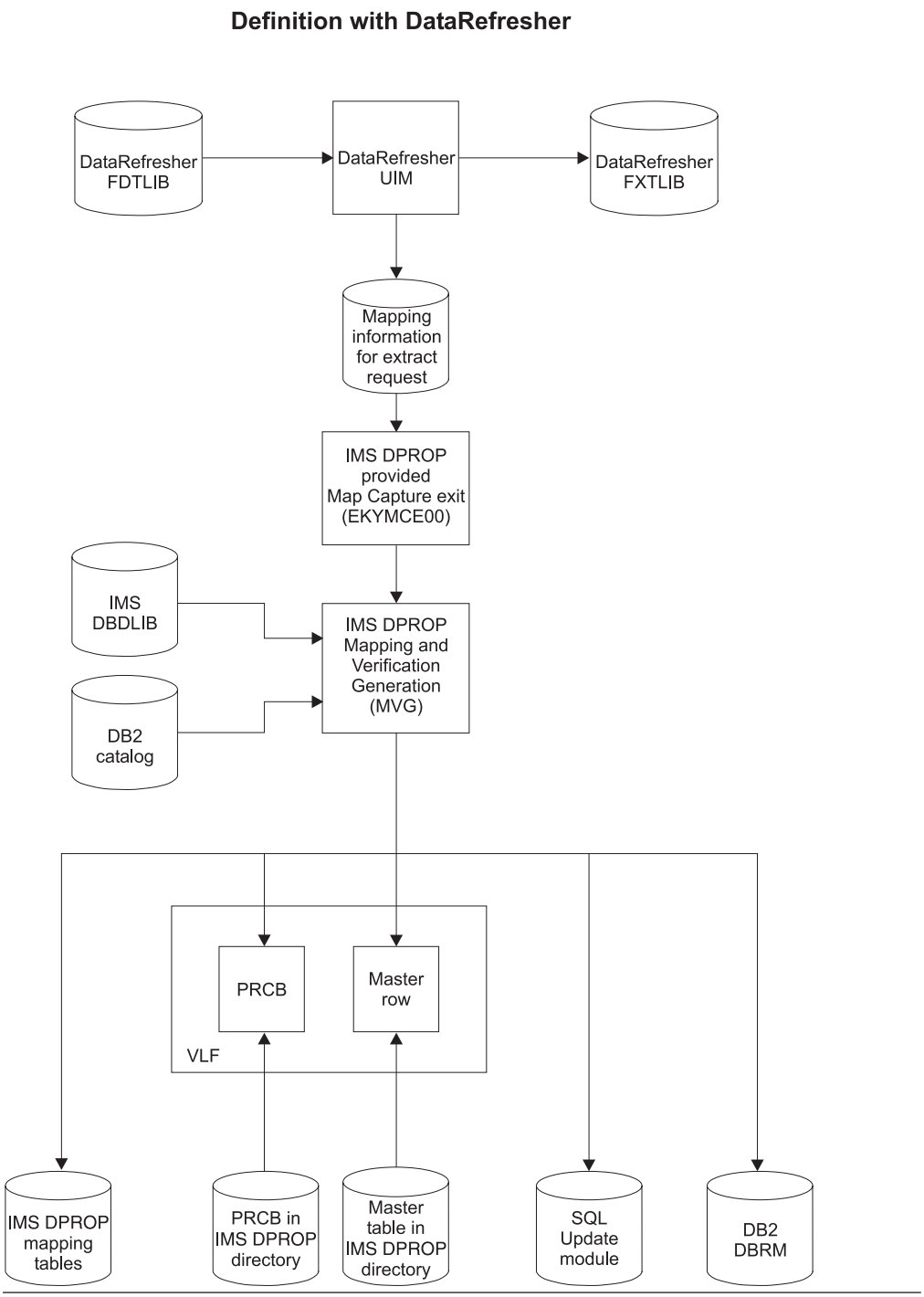


Figure 23. Propagation Request Definition with DataRefresher

DataRefresher and User Mapping Cases

For some segments, your mapping requirements might not be satisfied by the generalized mapping logic of IMS DPROP. Usually, such segments are propagated with Propagation exit routines. Propagation exit routines perform mapping, data conversions, and SQL updates during IMS-to-DB2 propagation and IMS updates during DB2-to-IMS synchronous propagation.

For user mapping cases, you might consider using the mapping and conversion capabilities of DataRefresher for extracts so that you don't have to write extract programs. The mapping and conversion done by DataRefresher should be compatible with the mapping and conversion done by your Propagation exit routine. Otherwise, you need to provide your own extract programs.

When determining whether to use DataRefresher to extract propagation requests belonging to a user mapping case, be aware that DataRefresher supports the following mapping capabilities:

- Nesting of internal segments and repeating groups of fields, so that an internal segment can contain, in turn, other internal segments
- Joining data from multiple IMS databases

If you need more information about the mapping capabilities of DataRefresher, or DXT refer to the appropriate library. Sample DataRefresher definitions for Propagation exit routines are discussed in the *IMS DataPropagator Reference*.

Defining Propagation Requests Using the MVG Input Tables

IMS DPROP provides an ISPF/TSO interface that you can use to define, update, and delete propagation requests stored in the MVG input tables. However, this section assumes you are not using the ISPF/TSO interface but instead are using SQL statements to build propagation requests in the MVG input tables.

The five MVG input tables are:
- PR table—propagation request table (DPRIPR)
- TAB table—target DB2 table (DPRITAB)
- SEG table—IMS segment table (DPRISEG)
- FLD table—IMS field table (DPRIFLD)
- WHR table—WHERE table (DPRIWHR), containing the WHERE clause of the propagation request

For more details on the MVG input tables and their columns, refer to the *IMS DataPropagator Reference*.

To define a propagation request for a generalized mapping case, you must provide a row in DPRIPR, one or more rows in DPRISEG, one row in DPRITAB, and one or more rows in DPRIFLD. If defining a propagation request with a WHERE clause, you must also provide one or more rows in the DPRIWHR table.

To define a propagation request for user mapping, you must provide at least one row in DPRIPR, DPRITAB, and DPRISEG.

This section covers:
- Identifying the propagation request
- Specifying the IMS segments to be propagated
- Specifying the DB2 tables
- Specifying the fields
- Executing the MVGU

Identifying the Propagation Request

DPRIPR contains one row of information for each propagation request being defined. Information includes the PR ID, which is also stored in all other tables of the MVG input tables. Storing the PR ID in all MVG input tables lets rows of the MVG input tables be identified as belonging to a specific propagation request. The

PR ID is also used as the name of the SQL update module that IMS DPROP generates whenever the propagation request is defined with MAPDIR=HR or TW and uses one of the generalized mapping cases.

Specifying the IMS Segments to be Propagated

You must identify the segments to be propagated by the propagation request being defined. If you are using generalized mapping cases, DPRISEG must contain a row for the entity segment and for each physical ancestor of the segment to be propagated, up to and including the root segment. If you are using mapping case 2, additional rows must exist for extension segments.

If you are using mapping case 3 and propagating an embedded structure, DPRISEG must also contain one row for each structure embedded in the segment. Embedded structures are referred to as internal segments.

If you are using a Propagation exit routine for user mapping, DPRISEG must contain a row for each segment to be propagated by the propagation request being defined. At IMS-to-DB2 propagation time, IMS DPROP calls the Propagation exit routine every time one of the segments is updated. For DB2-to-IMS synchronous propagation, IMS DPROP calls the Propagation exit routine each time a propagated table identified in the DPRITAB table is being updated.

Specifying the DB2 Tables

DPRITAB must contain one row for each DB2 table propagated by the propagation request. Generalized mapping allows only a single propagated table for each propagation request. A propagated table can have either a fully qualified or unqualified name.

Specifying the Fields

DPRIFLD must contain one row for each propagated field defined in the propagation requests. A given field in a segment can be either selected or not selected for propagation. If a field is selected for propagation, it must have a corresponding target column. For a non-selected field, the column name is left blank.

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 or to a particular segment. Mapping information includes the displacement, length, and format of the IMS fields to be propagated, as well as the DB2 table name, mapping case, and error option. For performance reasons, RUP PRCBs are located in both the IMS DPROP directory and in VLF.
- One HUP PRCB is created for each DB2 table synchronously propagated by a PRTYPE=E or U specifying MAPDIR=RH or TW. The PRCB contains mapping information for all propagation requests propagating from or to a table. 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, HUP PRCBs are located in both the IMS DPROP directory and in VLF.
- MVG updates the master timestamp field in the master table in the IMS DPROP directory to signal changes to RUP and HUP. If RUP, or HUP for synchronous, detects changes to the directory, it refreshes directory objects that are stored in memory so the changes become effective. The unique row of the master table is also stored in VLF to improve performance.
- For propagation requests belonging to a generalized mapping case and specifying MAPDIR=HR or TW, MVG also generates the assembler source code for an SQL update module. This module contains all the SQL update statements required to propagate data from IMS to DB2 based on propagation request definitions. IMS DPROP pre-compiles, assembles, and links the SQL source into a load library as an SQL update module. You must then bind the DBRM of the SQL update module either into a DB2 package or the plans of propagating applications before it can be used in propagation. Use a DB2 BIND operation to bind. You might want to bind the DB2 package automatically by using MVG.
- A flag in the propagation request in the MVG input table is set to indicate that the MVGU has processed the propagation request and placed it in the IMS DPROP directory.

The propagation request definition process using the MVG input tables is illustrated in Figure 24 on page 131.

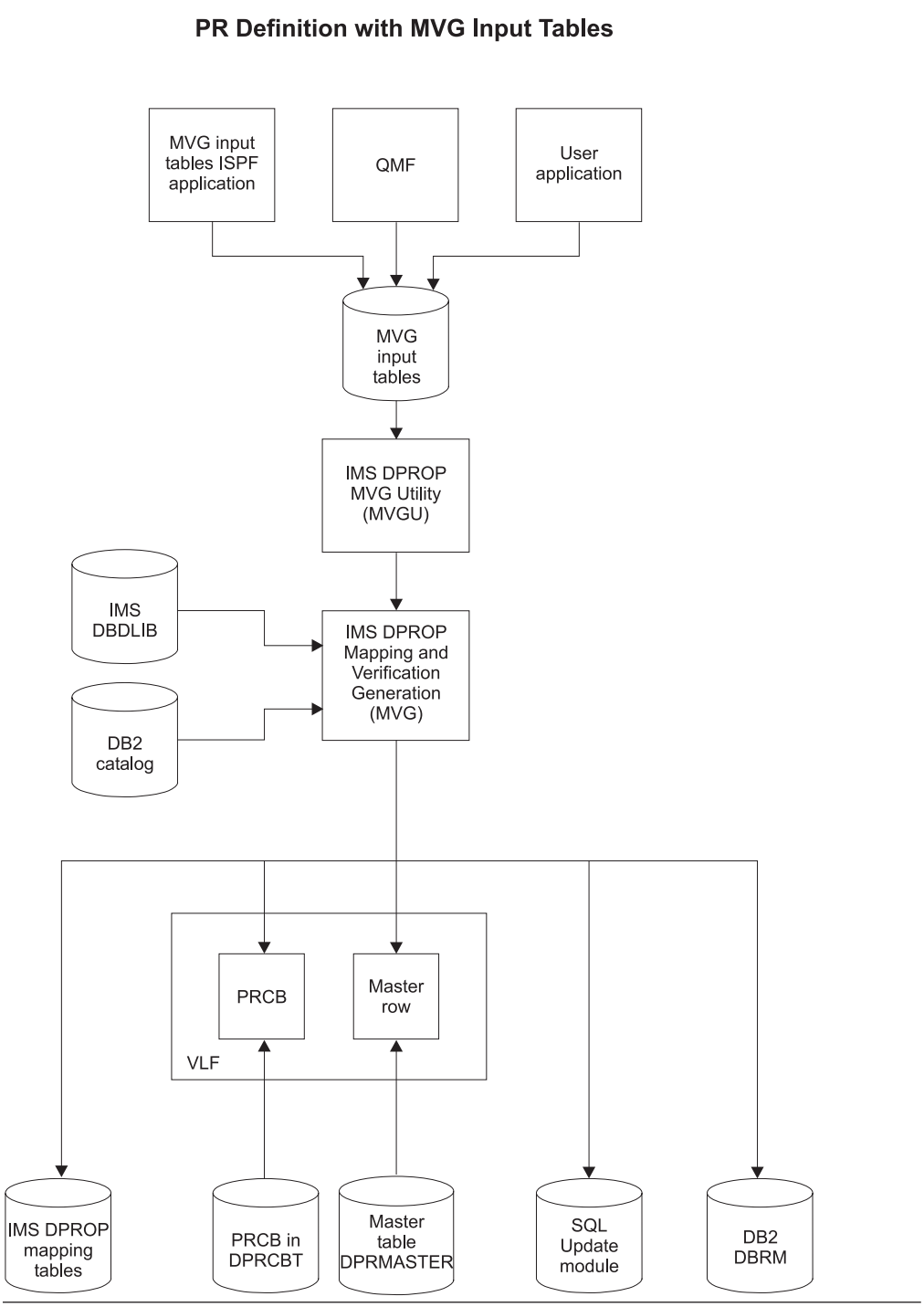


Figure 24. Propagation Request Definition with MVG Input Tables

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
- How to handle extension segments during DB2-to-IMS synchronous propagation when all source columns contain default values or null values
- 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
- PCB label, used only for DB2-to-IMS synchronous propagation
- Package bind options

This section briefly describes each parameter. For detailed information on these parameters and how to specify them, refer to the *IMS DataPropagator Reference*.

For additional considerations for LOG-ASYNC propagation, see *IMS DataPropagator Administrator's Guide for Log Asynchronous Propagation*.

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.

RH Relational to hierarchical only. For DB2-to-IMS synchronous propagation only.

TW Two-way synchronous propagation from IMS-to-DB2 and from DB2-to-IMS.

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 51.

ERROPT—Error Option

Specifies the error option (BACKOUT or IGNORE) to be taken when a propagation request fails. More information on this parameter is given in "IMS DPROP Error Option" on page 178.

MAXERROR—Maximum Number of Reported Propagation Errors

Specifies how many propagation failures for a propagation request are to be reported to the console and the audit trail. This parameter applies only when ERROPT=IGNORE is specified.

ACTION

Specifies whether the propagation request is to be added or replaced.

PRSET—Propagation Request Set Name

Specifies the set of propagation requests (PRSET) to which a single propagation request belongs. For more information, refer to the *IMS DataPropagator Reference*.

PROPSUP—Propagation Suppression

Specifies whether RUP and HUP should accept or reject a return code of 8 from a Segment exit routine. The Segment exit routine can use a return code of 8 to request suppression of propagation. For IMS-to-DB2 propagation, it is recommended that you suppress propagation by defining a WHERE clause during propagation request definition, when possible. For more information, refer to the *IMS DataPropagator Reference*.

AVU—Avoid Unnecessary Updates

Specifies whether IMS DPROP, when replacing a segment, should determine if at least one propagated field has changed. The parameter lets you influence the performance of IMS-to-DB2 propagation.

If you set AVU to YES, a replaced IMS segment results in a propagating SQL update only if at least one propagated field has changed. IMS DPROP compares the before-image and after-image of the replaced segment to determine whether any propagated field has changed. The path length is increased for the comparison, especially if the mapping involves a Segment exit routine, which is be called twice. But you avoid issuing unnecessary SQL update statements when no propagated field has changed. AVU set to YES can reduce the total IMS DPROP path length when only a subset of segment fields are propagated.

For DB2-to-IMS synchronous propagation, IMS DPROP always compares the before- and after-image and, therefore, always uses AVU=Y.

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

DEFVEXT—Default Value Extension Segments: Mapping Case 2 DB2-to-IMS Only

Tells IMS DPROP how to handle an extension segment during DB2-to-IMS synchronous propagation of an SQL insert and replace when all source columns of an extension segment contain null or default values.

If you specify DEFVEXT=N (NO), propagation of the SQL update or insert results in zero occurrences of the extension segment type. Specify DEFVEXT=N if you do not want to have extension segments that are mapped exclusively from columns having default or null values.

If you specify DEFVEXT=Y (YES), propagation of the SQL insert or update results in an occurrence of the extension segment type, even if all its fields are propagated from columns having a null or default value. DEFVEXT=Y is the default.[11]

Note that an SQL replace operation affects an extension segment only if at least one column mapped to the extension segment changes its value.

KEYORDER—DB2 Key Ordering Sequence

Specifies whether the columns of the DB2 primary key of the propagated table are ordered by the primary key index in ascending or descending sequence or whether MVG must access the DB2 catalog to determine the ordering sequence of each column. KEYORDER applies to all columns used in the primary key. If you have both ascending and descending columns used in the key, you must specify KEYORDER=ANY. Depending on what you specify you might have lengthy accesses to the DB2 catalog to determine the key order of each column.

PERFORM—Type of Operation: DataRefresher only

Specifies whether IMS DPROP and DataRefresher are to:

- Create a propagation request and store the extract request in EXTLIB (BUILDRUN)
- Create a propagation request without storing the extract request (BUILDONLY)
- Store the extract request only (RUNONLY)

You can run extract requests stored in EXTLIB using the DataRefresher DEM.

EXITNAME—Name of Propagation Exit

When you propagate using a Propagation exit routine (PRTYPE=U), specifies the name of the exit routine.

PROPSEGM—Propagated Segments: User Mapping with DataRefresher Only

Identifies the segments that are to be propagated by the propagation request being defined. At IMS-to-DB2 propagation time, the Propagation exit routine is called when one of the identified IMS segments is updated.

For DB2-to-IMS synchronous propagation, IMS DPROP calls the Propagation exit routine for the propagation request each time a propagated table identified in the DataRefresher EXTRACT statement or in the DPRITAB table is updated.

---

11. For DATE, TIME, and TIMESTAMP columns, IMS DPROP does not distinguish between the default and non-default values. Therefore, when processing DATE, TIME, and TIMESTAMP columns that are not null values, IMS DPROP assumes that they have a non-default value.

PCBLABEL—Label of IMS PCB for DB2-to-IMS Propagation Only

Specifies the label or name of the PCB that you create and reserve for HUP use in the IMS PSB. The IMS PCB is used by the generalized mapping logic of HUP to issue the IMS calls that propagate DB2 changes. Usually, this PCB is generated in the PSB with the LIST=NO keyword so that the PCB is transparent to your application programs.

If you are doing user mapping with Propagation exit routines, you can use PCBLABEL to identify a PCB that your Propagation exit routine can use.

PCBLABEL is only applicable for DB2-to-IMS synchronous propagation.

For more information, refer to "Defining the PCBs Reserved for HUP (DB2-to-IMS Synchronous Propagation)" on page 112.

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 Reference*.

Replacing a Propagation Request

To change a propagation request in the IMS DPROP directory, the propagation request must be recreated by using either DataRefresher or the MVG input tables in the same process used to create the original propagation request. SQL statements should *not* be used to change a propagation request in the IMS DPROP directory.

Ensure that the ACTION propagation parameter in the MVGPARM parameter is specified as REPL. Also, set the propagation request to INACTIVE state by running the SCU:

- If you are using DataRefresher to recreate the propagation request, change the DataRefresher statements as desired and then rerun DataRefresher UIM. If you want to change the propagation request without creating an extract request, use the propagation parameter PERFORM=BUILDONLY.
- If you are using the MVG input tables, use SQL to change input table information. You must set the PROCSED column of DPRIPR to either blank or N to indicate that the changed propagation request should be processed. Then you can issue the MVGU CREATE statement to update a propagation request in the IMS DPROP directory.

For additional information on changing a propagation request, see the *IMS DataPropagator Reference*.

Rebuilding a Propagation Request

You can use the MVGU RECREATE function to rebuild propagation control blocks in the IMS DPROP directory. The RECREATE function can also rebuild SQL update modules if they are destroyed. MVG retrieves information from the mapping tables and uses the information to recreate the objects for:

- Specific propagation requests
- Propagation requests propagating specific segments or databases
- All propagation requests defined in the mapping tables of the IMS DPROP directory

When processing RECREATE, MVGU does not access the MVG input tables, only the IMS DPROP directory. The RECREATE function does *not* alter the propagation requests stored in the mapping tables of the IMS DPROP directory.

For detailed information on how to code the RECREATE control statement, see the *IMS DataPropagator Reference*.You might also want to refer to *IMS DataPropagator Administrator's Guide for Log Asynchronous Propagation*.

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 8. Granting Privileges and Authorizations for DB2 Objects

This chapter discusses DB2 privileges in a data propagation environment. You must grant DB2 privileges or authority for different types of objects. This chapter distinguishes between granting privileges for:

- IMS DPROP tables, IMS DPROP utilities, and related objects
- Your propagated tables, your propagating applications, and associated objects

For IMS DPROP tables, utilities, and related objects, this chapter describes:

- Granting privileges for IMS DPROP directory tables, the audit trail table, and the MVG input tables
- If you use the DB2 package bind facility, binding the packages of IMS DPROP modules accessing IMS DPROP tables
- If you use the DB2 package bind facility, granting privileges for the two collection IDs containing the packages of:
  - IMS DPROP modules reading IMS DPROP tables
  - IMS DPROP utility modules updating IMS DPROP tables
- Binding the DB2 plans of IMS DPROP utilities
- Running IMS DPROP utilities

For your propagated tables, propagating applications, and related objects, this chapter describes:

- Granting privileges for your propagated tables
- If you use the DB2 package bind facility, granting privileges for the collection IDs containing the packages of SQL update modules and exit routines updating your propagated tables
- If you use the DB2 package bind facility, binding packages of SQL update modules and exit routines accessing the propagated tables
- Binding DB2 plans of propagating application programs
- Running propagating application programs

DB2 security mechanisms are very flexible, so you can establish DB2 privileges and authority many ways. This chapter provides general recommendations and describes only one of the many ways to establish DB2 security for data propagation.

To understand this chapter, you need to be familiar with DB2 security mechanisms. For more information on DB2 security, see *DB2 Administration Guide*.

IMS DPROP Tables, Utilities, and Related Objects

This section describes privileges and authority related to IMS DPROP tables, utilities, and related objects. Topics included in this section are:

- Granting privileges for IMS DPROP tables
- Binding packages of IMS DPROP modules
- Granting privileges for IMS DPROP collections
- Binding plans of IMS DPROP utilities
- Granting privileges for running IMS DPROP utilities

Granting Privileges for IMS DPROP Tables

You need to secure the following DB2 tables:
- IMS DPROP directory tables
- MVG input tables
- Audit trail table

IMS DPROP Directory Tables

Generally, the only people who should have privileges granted to them beyond SELECT for the IMS DPROP directory are those who own DB2 packages or DB2 plans used by IMS DPROP utilities. This prevents inadvertent updates to the IMS DPROP directory tables.

You can grant the SELECT privilege to PUBLIC for the following tables:
- DPRMASTER
- DPRCBT
- DPRHCBT
- DPRPR
- DPRWHR
- DPRMSG
- DPRSEG
- DPRTAB
- DPRFLD
- DPRRCT
- DPRPRCT
- DPRPRDSR
- DPRDRDSV

You should only update the directory tables with IMS DPROP utilities, MVG, and SCU. Do not use your own applications or QMF to insert, update, or delete rows in these tables. If you do so, the tables can contain erroneous or inconsistent control blocks, and IMS DPROP could generate unpredictable results.

MVG Input Tables

If you define all your propagation requests using DataRefresher, then you do not need to grant any privileges or even build the MVG input tables.

If you are using the MVG input tables to build propagation requests, then you need to grant the SELECT, UPDATE, INSERT, and DELETE privileges to the authorization identifiers used by people who:
- Build propagation requests in the MVG input tables
- Own the DB2 packages or plan of the MVG

The MVG input tables include:
- DPRIPR—propagation request table (PR table)
- DPRIWHR—WHERE clause table (WHR table)
- DPRITAB—target DB2 table (TAB table)
- DPRISEG—IMS segment table (SEG table)
- DPRIFLD—IMS field table) (FLD table)

Audit Trail Table

The audit trail table has three levels of privileges:
- SELECT privileges for people querying the audit trail table
- SELECT, UPDATE, INSERT, and DELETE privileges for people maintaining the audit trail table (for example, deleting old or outdated rows)

- SELECT, UPDATE, INSERT, and DELETE privileges for people owning the packages or plan of the AUDU utility

Binding Packages of IMS DPROP Modules

During IMS DPROP installation, you specify whether you intend to use the DB2 package bind facility. If using the facility, you identify two collection IDs for each IMS DPROP system. The collection IDs are referred to as the "IMS DPROP collections."

The IMS DPROP installation process binds the packages of IMS DPROP modules into these two collection IDs.

- The first IMS DPROP collection is used to bind packages of IMS DPROP utility modules reading and updating the IMS DPROP directory tables. It is called the"read-write IMS DPROP collection."
- The second IMS DPROP collection is used to bind packages of IMS DPROP modules reading IMS DPROP directory tables. It is called the "read-only IMS DPROP collection."

The authorization ID you use to do IMS DPROP installation must have the following privileges:

- BINDADD and CREATE IN COLLECTION privilege, for binding new packages, or BIND privilege, if binding again an existing package
- SELECT, UPDATE, INSERT, and DELETE privileges for the IMS DPROP directory tables, MVG input tables, and audit trail table
- SELECT privilege for the DB2 catalog tables, needed because some IMS DPROP modules read information from the DB2 catalog

Binding plans using package bind are further discussed in Chapter 9, "Binding and Administering Plans," on page 149.

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 141.

When granting the privileges:

- Be restrictive when granting the CREATE IN COLLECTION privilege. Usually, only the IMS DPROP system administrator needs to bind into these collections packages of IMS DPROP modules. Therefore, only an authorization ID used by the system administrator needs these privileges for these collections.
- Be restrictive when granting the EXECUTE privilege for the read-write IMS DPROP collection. Usually only the IMS DPROP system administrator needs to bind and own the plans of IMS DPROP utilities. Therefore, only an authorization ID used by the system administrator needs the EXECUTE privilege for these collections.
- You do not need to be restrictive when granting the EXECUTE privilege for the read-only IMS DPROP collection. All owners of DB2 plans of propagating applications and IMS DPROP utilities need the EXECUTE privilege. Since the packages of this collection provide read-only access, you might want to grant the EXECUTE privilege to PUBLIC.

Also consider granting BIND and COPY privileges for the two IMS DPROP collections. Be restrictive when granting these privileges. Usually only IMS DPROP system administrators need these privileges.

Binding Plans of IMS DPROP Utilities

During IMS DPROP installation, you should bind the DB2 plans of the following IMS DPROP utilities:

- AUDU
- CCU
- MVGU
- SCU
- DLU

If you use the DB2 package bind facility, binding these plans requires the following authorizations:

- BINDADD privilege, for binding new plans, or BIND privilege, if binding again an existing plan
- EXECUTE privilege for the DB2 collection IDs containing IMS DPROP packages

If you do not use the DB2 package bind facility, binding the IMS DPROP utility plans requires the following authorizations:

- BINDADD privilege, for binding new plans, or BIND privilege, if binding again an existing plan
- SELECT, UPDATE, INSERT, and DELETE privileges for the IMS DPROP directory tables, MVG input tables, and audit trail table
- SELECT privilege for DB2 catalog tables needed because some IMS DPROP modules read information from the DB2 catalog

After binding the plans for the utilities, you need to grant the EXECUTE privilege for them. Usually, this privilege is granted to authorization IDs used by systems programmers, database administrators, and operations personnel.

Binding plans using package bind are further discussed in Chapter 9, “Binding and Administering Plans,” on page 149.

Running IMS DPROP Utilities

Running an IMS DPROP utility requires the EXECUTE privilege for the DB2 plan of the utility. Execution of some IMS DPROP utilities requires *additional* privileges, as described in this section:

- Additional authorizations required to execute CCU
- Additional authorizations required to execute DLU
- 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 Execute DLU

The DLU reads the rows of the propagated tables with dynamic SQL statements. Therefore, the person who runs the DLU needs the SELECT privilege for the propagated tables.

Additional Authorizations Required to Run MVG/MVGU

When creating or re-creating propagation requests for a generalized mapping case for IMS-to-DB2 propagation, MVG creates an SQL update module. As an option, MVG does an automatic package bind of the DBRM of the SQL update module into the collection ID that you specify.

To use the MVG bind option, the person who runs the MVG must have either the SYSADM or the SYSCTRL privilege or must be granted all the following privileges:

- BINDADD and CREATE IN COLLECTION privilege for the collection ID where the package of the SQL update module is bound, for binding new packages, or BIND privilege for the package, if re-binding an existing package.
- SELECT, UPDATE, INSERT, and DELETE privilege on the DB2 propagated table affected by the propagation request. These privileges are also necessary for people having the SYSCTRL privilege.

When deleting a propagation request, MVGU can delete packages previously bound by MVG. For MVGU to delete a package, you must own the package, have the package owner grant you the BINDAGENT privilege, or you must be granted SYSCTRL or SYSADM authority.

Additional Privileges Required to Execute the SCU

Various DB2 privileges must be granted to execute the following SCU control statements:

- ACTIVATE, DEACTIVATE, and SUSPEND control statements for propagation requests for DB2-to-IMS synchronous propagation. The SCU must ensure affected DB2 propagated tables are not concurrently updated. The SCU issues internally SQL LOCK TABLE statements and DB2 DISPLAY DATABASE commands to check or concurrent updates. Issuing SQL LOCK TABLE statements requires at least one of the following privileges:
  - SYSADM or SYSCTRL authority
  - DBADM authority for the database
  - Ownership of the table
  - SELECT privilege for the table

  Issuing the DB2 DISPLAY DATABASE command also requires privileges, such as the DB2 DISPLAY privilege. Therefore, the person who runs the SCU must be granted the appropriate privileges.
- ESTOP, RESET, INIT DPROP, INIT STATF control statements. When processing these control statements, the SCU updates the IMS DPROP status file. If you have protected the status file with RACF or an equivalent, the person who runs these SCU control statements needs RACF authorization to update the IMS DPROP status file.
- 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, DLU, and MVGIN front end applications read the rows of the IMS DPROP directory tables or MVG input tables with dynamic SQL statements. Therefore, the person who 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, Propagating Applications, and Related Objects

This section describes privileges and authorization relating to:

- Propagated tables
- Propagating collections
- Binding packages of SQL update modules and Propagation exit routines
- SQL update modules bound into different packages
- DB2 plans of propagating applications
- Propagating applications

Granting Table Privileges for Propagated Tables

This section provides considerations for granting privileges for:

- One-way IMS-DB2 propagation
- DB2-to-IMS synchronous propagation
- Two-way synchronous propagation

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 resynchronizing the DB2 copy after propagation has been suspended

You should grant the SELECT privilege for propagated tables to people:

- Using decision support systems
- Querying propagated tables
- Running CCU
- Running DLU

Updates to Nonpropagated Columns: You can update nonpropagated 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 nonpropagated 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 25 illustrates the concept of updating nonpropagated columns.

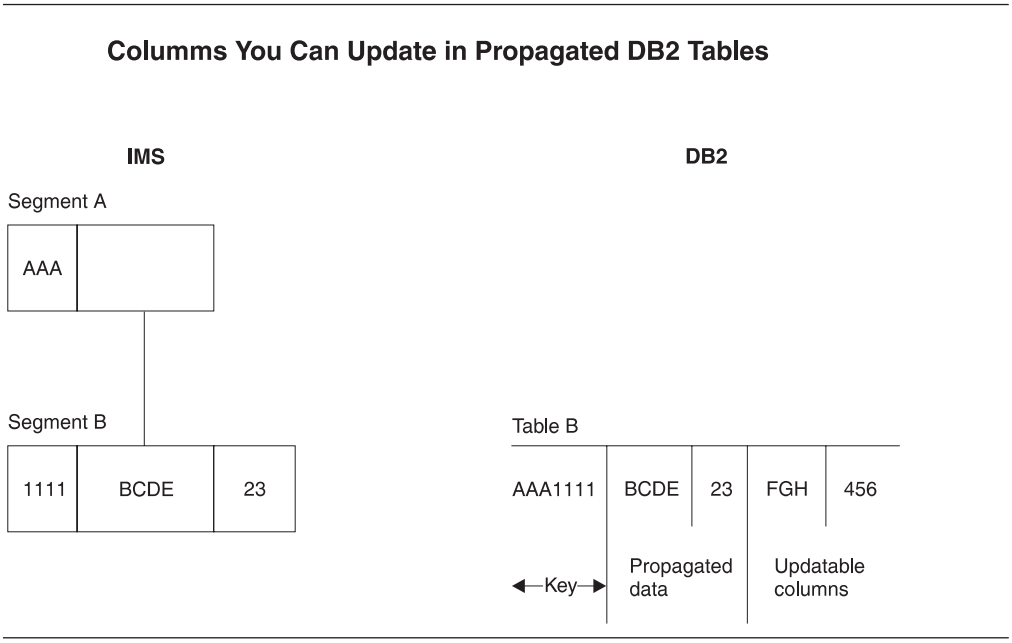


Figure 25. Columns That Can Be Updated in Propagated DB2 Tables. The columns containing FGH and 456 can be updated through a view. The key and propagated data should be read-only.

If your IMS and DB2 data become inconsistent, you might not be able to resynchronize data using a re-extract or a CCU repair file, because some of your columns are nonpropagated. You might have to provide a program of your own to resynchronize data.

DB2-to-IMS and Two-Way Synchronous Propagation

DB2-to-IMS and two-way synchronous propagation do not introduce special considerations for table privileges of propagated tables. However, you should protect propagated tables from nonpropagating SQL updates. The following types of SQL updates are *not* propagated:

- SQL updates issued by programs or tools that do not run in an IMS region and do not use DB2's IMS attachment facility
- Remote SQL updates issued from a Distributed Data Facility (DDF) connection
- SQL updates issued when tracing for monitor class 6 has not been started or has been inadvertently stopped

To prevent such updates, we recommend that you set the DB2 system parameter DPROP SUPPORT to 2. See "Preparing DB2 for Data Propagation for DB2-to-IMS Propagation" on page 114. If your environment prevents you from using the parameter, consider using DB2 validation procedures and restricting privileges to stop trace monitoring for class 6. For details on this subject, see "Protecting Propagated Tables from Nonpropagating SQL Updates" on page 117. Also grant the SELECT privilege to authorization IDs that run CCU and DLU.

If doing two-way propagation, you should also grant the SELECT, UPDATE, INSERT, and DELETE privileges to authorization IDs that become owners of the:

- Packages of SQL update modules and propagation exit routines, if using the DB2 package bind facility
- Plans of propagating IMS applications, if you do not use the DB2 package bind facility

Granting Privileges for Propagating Collections

If you want to use the DB2 package bind facility and do IMS-to-DB2 propagation, you need to bind the DBRMs of SQL update modules and Propagation exit routines into DB2 packages. You need to decide into which collections the packages should be bound. These collections are referred to as the"propagating collections."

You might want to use different propagating collections for production work and tests.

Consider the following DB2 privileges when propagating collections:

- The CREATE IN COLLECTION privilege must be granted to owners of the packages of SQL update modules and Propagation exit routines. This privilege is required for binding the packages.
- Consider granting the EXECUTE privilege for the *entire* collection ID, instead of individual packages, to the owners of plans of propagating applications. The EXECUTE privilege is required to bind the plans of propagating applications.

  If you do not grant the EXECUTE privilege for the entire collection ID to future owners of plans of propagating applications, you must grant the EXECUTE privilege for individual packages.

  Granting the EXECUTE privilege for the *entire* collection simplifies administration of DB2 plans of your propagating applications. It allows you, when binding the plans of propagating applications, to specify PKLIST(collection.*) instead of explicitly identifying each required package. You do not need to know which package is required for which plan.
- You also need to decide if you should grant the BIND and COPY privileges for the entire collection ID or for individual packages.

Binding Packages of SQL Update Modules and Propagation Exit Routines

If you want to use the DB2 package bind facility and do IMS-to-DB2 propagation, you bind the DBRMs of SQL update modules and Propagation exit routines into DB2 packages. You can bind the DBRMs of SQL update modules as part of MVG processing when you create or recreate the propagation request.

The bind process requires that the owner of the DB2 package have the following privileges:

- BINDADD and CREATE IN COLLECTION privilege, for binding new packages, or BIND privilege for the package, if binding an existing package again
- CREATE IN privilege for the collection ID
- SELECT, UPDATE, INSERT, and DELETE privileges for the propagated tables

If you do not grant the EXECUTE privilege for the entire collection ID to future owners of plans of propagating applications, you must grant the EXECUTE privilege for individual packages. You might also need to grant the BIND and COPY privilege for individual packages.

Chapter 9, “Binding and Administering Plans,” on page 149 has additional information about binding packages.

Binding SQL Update Modules into Different Packages

If you have defined your propagation requests with unqualified table names, then you will often want to bind the DBRM of the SQL update module into different packages using different table-name qualifiers. You can do this using the COPY and QUALIFIER keywords of the DB2 BIND command.

A BIND with the COPY option uses a previously-bound package of the SQL update module as input and creates a new package accessing the propagated tables with the specified QUALIFIER keyword.

The BIND COPY process requires that the owner of the new package have the following privileges:

- COPY privilege for the package, or its collection, being copied
- BINDADD privilege and BIND privilege for the package or collection
- CREATE IN privilege for the collection ID
- SELECT, UPDATE, INSERT, and DELETE privileges for the propagated tables

If you do not grant the EXECUTE privilege for the entire collection ID to future owners of plans of propagating applications, you must grant the EXECUTE privilege for individual packages.

Chapter 9, “Binding and Administering Plans,” on page 149 has additional information about binding packages.

Binding DB2 Plans of Propagating Applications

You must bind DB2 plans for propagating applications and for the receiver programs used in user asynchronous propagation. This requirement stems not only from the SQL updates made by IMS DPROP, but also from SQL reads of the IMS DPROP directory, which is composed of DB2 tables.

To bind the plans, the plan owner must have the following privileges if using the DB2 package bind facility:

- BINDADD privilege for binding new plans, or BIND privilege for the plan if binding an existing plan again
- EXECUTE privilege for the read-only IMS DPROP collection ID
- EXECUTE privilege for the propagating collections or for the packages of individual SQL update modules and Propagation exit routines

If you are not using DB2 package bind, the plan owner must have the following privileges to bind the plans of propagating applications:

- BINDADD privilege for binding new plans, or BIND privilege for the plan if binding an existing plan again
- SELECT privilege for the IMS DPROP directory tables
- SELECT, UPDATE, INSERT, and DELETE privileges for the propagated tables

  After binding the plans of the propagating programs and receiver program, you need to grant the EXECUTE privilege for them. See “Running Propagating Applications” on page 148.

Chapter 9, “Binding and Administering Plans,” on page 149 has additional information about binding DB2 plans.

Running Propagating Applications

Users of propagating application programs or receiver programs must have the EXECUTE privilege for these plans. You should review DB2 Administration Guide to determine the method of implementing DB2 security that best suits your needs. The techniques described in the following sub-sections are suggestions for:

- Message processing and Fast Path regions
- IMS batch and batch message processing programs
- DB2 sign-on authorization exits

Message Processing and Fast Path Regions

Authorization for IMS MPPs, message-driven BMPs, and Fast Path regions is usually done using IMS transaction code security. Authorization for related DB2 plans can be granted to PUBLIC. Therefore, authorization to execute the transaction can be viewed as authorization to execute the plan. This method also reduces overhead required by DB2 authorization processing.

If you are granting the EXECUTE privileges of plans of MPPs/IFPs to PUBLIC, consider preventing misuse of these plans in environments other then message processing and Fast Path regions. Specify the ENABLE keyword in the DB2 BIND command. For example, you can use the ENABLE keyword to limit the execution of a plan to the message regions of a specific IMS online system. For example:

```
BIND PLAN (planname) ... ENABLE (IMSMPP) IMSMPP (imsid)
```

You can also grant authorization to functional identifiers. Functional identifiers identify functional groups, such as an accounts payable department, to the DB2 system. If you use functional identifiers, you need a DB2 Sign-on Authorization exit routine. The exit routine associates user identifiers that need to execute propagating programs with functional identifiers.

IMS Batch and Batch Message Processing Programs

You can grant authorization for batch and BMP programs to specific functional identifiers or to specific user IDs. If functional identifiers are used, you need a DB2 Sign-on Authorization exit routine.

DB2 Sign-on Authorization Exits

You can use a DB2 Sign-on Authorization exit routine to associate user identifiers with functional identifiers. If you are using this exit routine, you can grant the EXECUTE privilege for the DB2 plans of propagating applications to the functional identifiers. This minimizes the number of identifiers to which authorization to execute the DB2 plan must be granted; it also reduces the administrative efforts required to maintain authorizations.

Chapter 9. Binding and Administering Plans

This chapter describes:

- Binding DB2 plans of propagating applications
- Administering DB2 plans with and without a Resource Translation table (RTT)

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. Bind plans for Receivers, propagating application programs, and receiver programs. You must also be authorized to use the plans.

You can bind plans with or without use of the DB2 package bind function.

Binding Plans with Bind Package

With DB2 V3 R1 and following releases, you can use the DB2 package bind facility for the DB2 plans of your propagating applications and asynchronous receiver program.

Using the DB2 package bind facility, you can bind an individual DBRM as a *package* into a *package collection*, identified by a *collection ID*. Then, when binding the DB2 plan, you specify which packages, collection IDs, and DBRMs will be included in the DB2 plan.

Using the DB2 package bind facility has the following advantages:

- You do not need to perform another bind for the DB2 plans of propagating applications when propagation requests affecting those applications are changed or added. Instead, you only need to bind the DBRM of the affected SQL update module into a package. You reduce the number of required bind operations and simplify administration of DB2 plans.

  For example, using bind package you do not need to do both an initial and a subsequent bind for the plans of propagating applications.
- The bind for individual plans is simplified, because you do not need to specify a complete list of DBRMs that are part of the DB2 plan. You do not need to keep track of which DBRM is required in which plan. Instead, when binding a plan, you can specify the collection IDs.
- You can benefit from having different ISOLATION attributes for different packages. Packages of IMS DPROP modules should be bound with the ISOLATION level *cursor stability* (CS) to reduce the chance of DB2 enqueue conflicts on the small IMS DPROP directory tables. If applications that issue SQL statements require it, you can still bind the packages for your application modules with the ISOLATION level *repeatable read* (RR).
- Bind package offers more flexibility to qualify table names of static SQL statements that have unqualified table names. When application program runs, different modules can issue static SQL statements with unqualified table names. For example, SQL statements with unqualified table names can be issued by IMS DPROP modules accessing the IMS DPROP directory, by SQL update modules and Propagation exit routines updating the propagated tables, and by your own application modules accessing their own 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.
- One or more collections for application modules issuing SQL calls.

Determine which package collection is used for each purpose and determine the collection IDs used to bind packages and plans. Also grant the following DB2 privileges for package collections:

- CREATE IN COLLECTION privilege for each collection. This privilege is needed to bind a package into a specific collection.
- EXECUTE privilege for entire collections. This privilege is needed to bind the plan of propagating applications if you are specifying entire collections on the PKLIST keyword of the BIND PLAN command, as recommended by IMS DPROP.
- Depending on the standards of your installation, possibly the BIND and COPY privileges for entire collections.

Job Stream for Binding DB2 Packages

If you are using the DB2 package bind option, you will usually bind packages required to run your propagating applications and the asynchronous receiver program, such as:

- Packages of IMS DPROP modules accessing IMS DPROP directory tables in read-only mode. These packages are usually bound into the IMS DPROP read-only collection during IMS DPROP installation.
- Packages of SQL update modules used with propagation requests belonging to generalized mapping cases and used for IMS-to-DB2 propagation. These packages are usually bound when propagation requests are created as part of MVG processing.

  If you create propagation requests with unqualified table names and use the same propagation request to propagate to multiple, identically structured tables

with different qualifiers, then you can use a BIND COPY command to bind additional packages with different qualifiers.

- Packages of your Propagation exit routines. These packages are bound after creation of the tables and precompilation and compilation of Propagation exit routines.
- Packages of application modules issuing SQL statements.

Figure 26 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 26. BIND PACKAGE Job Stream for IMS DPROP

Notes:

1. This is the collection ID where the package is to be bound. The owner of the package must be granted the CREATE IN privilege for this collection ID.
2. This is the name of the DBRM to be bound in a package.
3. This is the library containing the DBRM used as input to the bind package.
4. Specify the ISOLATION parameter as CS (cursor stability) or RR (repeatable read) depending on the needs of the module.

   IMS DPROP packages located in the read-only IMS DPROP collection should be bound with CS.
5. Specify the RELEASE parameter as COMMIT so that DB2 resources are released at commit time. The resources can then be used for concurrent processing.
6. Specify the VALIDATE parameter as BIND so that DB2 resources used by the package are validated when the package is bound, rather than when the application runs. This improves performance of your propagating programs, especially propagating MPPs and IFPs.
7. If you specify a QUALIFIER, its value is the qualifier for any unqualified names in static SQL statements that are present in the DBRM being bound. For example, use the QUALIFIER keyword when binding the package of an SQL update module of a propagation request created with an unqualified table name.
8. The OWNER keyword specifies the owner of the package being bound. If the OWNER keyword is not present, it defaults to the primary AUTHID of the bind process.

If the QUALIFIER keyword is not specified, then the owner is used as qualifier for any unqualified table name in static SQL statements of the DBRM being bound.

Job Stream for Binding DB2 Plans with Bind Package

Figure 27 is a sample job stream for binding a plan of a Receiver, a propagating application, or receiver program. The numbers in the figure correspond to the notes following the figure.

```
//jobname JOB
//BIND    EXEC    PGM=IKJEFT01
//SYSTSPRT  DD    SYSOUT=*
//SYSPRINT  DD    SYSOUT=*
//SYSUDUMP  DD    SYSOUT=*
//SYSTSIN   DD    *
 DSN SYSTEM(DSN)
 BIND PLAN (planname)                                  -
      ACTION(REPLACE)                                  -
      PKLIST(propag-colid2.*,        1                 -
             appl-colid2.*,          2                 -
             ...                                       -
             dprop-colid3.*)         3                 -
      LIBRARY('applpgm.dbrmlib1',    4                 -
              'applpgm.dbrmlib2')                      -
      MEMBER(appl_dbrm6              4                 -
             appl_dbrm7)                               -
      ISOLATION(CS)                  5                 -
      RELEASE(COMMIT)                5                 -
      ACQUIRE(USE)                   5                 -
      VALIDATE(BIND)                 5                 -
      QUALIFIER(qualifier)           5                 -
      OWNER(owner)                   6                 -
      ENABLE (IMSMPP) IMSMPP (imsid) 7                 -
      RETAIN
/*
```

Figure 27. BIND PLAN Job Stream When Using Packages

Notes:

1. *propag-colid2* identifies a propagating collection. In this example, there is only one propagating collection. It contains the packages of the SQL update modules of propagation requests for the Receiver, application, or receiver program. It also contains the packages of any user-written IMS DPROP exit routines that issue SQL calls.

   Depending on how your system is organized, you might need to provide collection IDs of more than one propagating collection in the PKLIST keyword.

   In this example, PKLIST identifies entire collection IDs rather than individual packages, as specified by coding a .* after the collection ID. Specifying the entire collection is convenient when you do not need to know which propagation request is used by each plan and application. However, specifying .* requires that the owner of the DB2 plan have the EXECUTE privilege on the entire collection.
2. *appl-colid2* is the name of a collection ID containing the packages of user-written application modules that issue SQL statements.
3. *dprop-colid3* identifies the IMS DPROP read-only collection. This collection contains the packages of IMS DPROP modules reading the IMS DPROP directory tables. Each IMS DPROP system has its own 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 receiver program, you either provide different ENABLE specifications or omit the ENABLE specifications.

Binding Plans without Bind Package

This section describes binding the plans of Receivers, application programs, and asynchronous receiver programs without using the DB2 package bind option. Topic in this section are:

- Binding the Receiver
- Binding synchronous propagation applications
- Binding the user asynchronous receiver program
- Job stream for binding DB2 plans without bind package
- DB2 ALIAS and SYNONYM statements

Binding Synchronous Propagation Applications

Some applications running under IMS may already access DB2 tables through DB2's IMS attachment facility. When these programs also become involved in propagation, the application plan consists of:

- DBRMs of IMS DPROP modules reading the IMS DPROP directory tables
- Original application DBRMs from the precompilation of the application modules

Initial Bind

When propagation is synchronous, binding the DB2 plans of propagating applications should be completed before activating IMS Data Capture, through the EXIT keyword in the IMS DBD, and DB2 Data Capture through, the DATA CAPTURE CHANGES option on the CREATE TABLE or ALTER TABLE statement. The *initial bind* is usually done before the creation of propagation requests.

You need to include the following during the initial bind process:

- DBRMs for IMS DPROP's read-access to the IMS DPROP directory tables
- DBRMs for the application's access to the DB2 tables, if any

Subsequent Bind

For IMS-to-DB2 propagation, if you do not use the DB2 package bind facility, you need to use BIND to re-bind the plans of your propagating applications after creating, replacing, or deleting a propagation request. During the subsequent bind process, include:

- DBRMs for SQL update modules of propagation requests run by the application, if performing IMS-to-DB2 propagation
- DBRMs for IMS DPROP exit routines issuing SQL statements
- DBRMs for IMS DPROP's access to the IMS DPROP directory tables
- DBRMs for the application's access of DB2 tables, if any

Binding the User Asynchronous Receiver Program

The DB2 plan for a receiver program for user asynchronous propagation consists of the DBRMs of IMS DPROP modules and the original DBRMs of the receiver program. If you are doing user asynchronous propagation, the receiver program that calls RUP needs to be bound with:

- DBRMs for SQL update modules for propagation requests
- DBRMs for IMS DPROP exit routines issuing SQL statements
- DBRMs for RUP's access to the IMS DPROP directory tables
- DBRMs for the receiver program's access to DB2 tables, if any

Job Stream for Binding DB2 Plans without Bind Package

This section applies to only synchronous and user asynchronous propagation.

Figure 28 on page 155 is a composite BIND job stream showing all DBRMs that might be required to bind an application or receiver program. The numbers in the figure correspond to the notes following the figure.

```
//jobname JOB
//BIND    EXEC    PGM=IKJEFT01
//SYSTSPRT  DD    SYSOUT=*
//SYSPRINT  DD    SYSOUT=*
//SYSUDUMP  DD    SYSOUT=*
//SYSTSIN   DD    *
DSN SYSTEM(DSN)
BIND PLAN (planname)                                    -
     LIBRARY('DPROP.EKYDBRM'                          -
             'applpgm.dbrmlib'        1                 -
             'sqlupdt.dbrmlib'                          -
             'exitr.dbrmlib'                            -
             'receiver.dbrmlib')                        -
     MEMBER(EKYX120X                  2                 -
            EKYGC001                  3                 -
            EKYGH001                                    -
            EKYGM001                                    -
            ...                                         -
            appl_dbrm1                4                 -
            appl_dbrm2                                  -
            ...                                         -
            pr1                       5                 -
            pr2                                         -
            ...                                         -
            prnn                                        -
            exitr1                    6                 -
            exitr2                                      -
            ...                                         -
            exitrnn                                     -
            receiver_dbrm1            7                 -
            receiver_dbrm2)                             -
     ISOLATION(CS)                    8                 -
     RELEASE(COMMIT)                  9                 -
     ACQUIRE(USE)                    10                 -
     VALIDATE(BIND)                  11                 -
     ACTION(REPLACE)                                    -
     QUALIFIER(qualifier)                               -
     OWNER(owner)                                       -
     ENABLE (IMSMPP) IMSMPP (imsid)  12                 -
     RETAIN
/*
```

Figure 28. BIND Plan Job Stream without Packages

Notes:

1. This is a concatenation of the IMS DPROP-supplied DBRM library and any user DBRM libraries associated with included DBRM members.
2. The EKYX120X DBRM name must be coded as shown. EKYX120X uses unqualified table names to reference IMS DPROP directory tables. The BIND process sets the qualifier of the IMS DPROP directory table names. If the qualifier set by BIND does not satisfy your requirements, you might want to use a DB2 CREATE SYNONYM or CREATE ALIAS statement for the propagated tables. See "DB2 ALIAS and SYNONYM Statements" on page 156 for more information.
3. The following DBRM names might vary depending on your installation:
   - EKYGC001
   - EKYGH001
   - EKYGM001

These DBRM names are for specific IMS DPROP systems you have defined. Each IMS DPROP system has its own copy of these modules. The first IMS DPROP system uses the suffix 001, the second IMS DPROP system uses 002, the third IMS DPROP system uses 003, and so on. You need to know which IMS DPROP system you are using to specify the appropriate suffix for these two DBRMs.

4. Application DBRMs are necessary if the application program issues any SQL calls.
5. For IMS-to-DB2 propagation, there are DBRMs for SQL update modules of all propagation requests for the application or receiver program. This does not apply to the initial bind.
6. There are DBRMs for any user-written exit routines that issue SQL statements. This does not apply to the initial bind.
7. Specify the ISOLATION parameter as CS (cursor stability) to reduce contention on the IMS DPROP directory tables and improve concurrent access to them.
8. Specify the RELEASE parameter as COMMIT to release DB2 resources at commit time; you can then use the resources for concurrent processing.
9. Specify the ACQUIRE parameter as USE so that DB2 locks and resources are acquired when used, instead of when allocated.
10. Specify the VALIDATE parameter as BIND so that DB2 resources used by the plan are validated when the plan is bound, instead of when the application begins.
11. In this example, the ENABLE keyword restricts use of the plan to IMS MPP programs of a specific IMS online system. The plan of an MPP is bound and the installation uses IMS transaction security and grants the EXECUTE privilege of the plan to PUBLIC to improve performance. Restricting use of the plans to MPPs prevents accidental and intentional misuse of the plan in environments that are not protected by IMS transaction security.

    When binding the plan of BMPs, batch programs, and the receiver program, you do not specify ENABLE.

DB2 ALIAS and SYNONYM Statements

You usually use DB2 aliases and synonyms in installations where the DB2 package bind is not used. The installations include DBRMs directly into their DB2 plans.

The following static SQL statements issued by IMS DPROP have unqualified table names:

- Most SQL statements issued to access IMS DPROP directory table DPRMASTER
- Propagating SQL statements issued by SQL update modules if you specify during propagation request definition unqualified table names for the propagated tables

During the bind process, the qualifier for these SQL statements is set based on either the:

- Authorization ID used for the bind process
- Authorization ID on the QUALIFIER keyword
- Optional OWNER keyword of the BIND command in DB2

Sometimes the qualifier set by BIND is not convenient. For example, the qualifier set by BIND might be different from the qualifier for your IMS DPROP directory tables or the propagated tables. Before the bind you can use a DB2 CREATE ALIAS or CREATE SYNONYM statement to match the bind and IMS DPROP qualifier.

Figure 29 shows the two-step BIND process when you create aliases or synonyms before bind.

Figure 29. 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 30.

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

Figure 30. Using the DB2 CREATE ALIAS Statement

In the example:
- The qualifier of the IMS DPROP directory tables is PROD
- The qualifier set by a bind process will be SANDY

Therefore, the CREATE ALIAS statement creates the alias SANDY.DPRMASTER for the PROD.DPRMASTER table.

When the bind of the plan of a propagating application sets SANDY as the qualifier for unqualified SQL statements, access to DPRMASTER is qualified as SANDY.DPRMASTER. If the alias is created before the bind, ALIAS processing translates SANDY.DPRMASTER into PROD.DPRMASTER. Therefore, unqualified IMS DPROP SQL statements for the DPRMASTER table access the PROD.DPRMASTER table.

If you define propagation requests with unqualified table names, you might also want to create aliases for the propagated tables.

Using the CREATE SYNONYM Statement

Before the bind process, you can use the DB2 CREATE SYNONYM statement to create a synonym for the DPRMASTER directory table. Use the unqualified name

of the IMS DPROP directory table, DPRMASTER, as the synonym. The authorization ID used to issue the CREATE SYNONYM statement should be the same as the qualifier later set by the bind process.

```
CREATE SYNONYM DPRMASTER FOR PROD.DPRMASTER
```

Figure 31. Using the DB2 CREATE SYNONYM Statement

In the example:
- The qualifier of the IMS DPROP directory tables is PROD
- The qualifier set by an eventual bind process will be SANDY

Therefore, the above CREATE SYNONYM statement should be issued by the authorization ID SANDY.

When the bind of the plan of a propagating application sets SANDY as the qualifier for unqualified SQL statements, access to DPRMASTER is qualified as SANDY.DPRMASTER. If you issue the CREATE SYNONYM statement before the bind by the authorization ID SANDY, then during the bind SYNONYM processing translates the qualified name SANDY.DPRMASTER into PROD.DPRMASTER. Therefore, unqualified IMS DPROP SQL statements for the DPRMASTER table access the PROD.DPRMASTER table.

If you define propagation requests with unqualified table names, you might also want to create synonyms for the propagated tables.

Administering DB2 Plans with or without a Resource Translation Table (RTT)

You can administer plans for online or batch regions in two ways:
- Use of an RTT so that one DB2 plan can be shared by multiple propagating application programs. The RTT associates application programs and plan names.
- Use of a different DB2 plan for each application program.

Use the method you currently have in place. If you are new to IMS/DB2 mixed-mode applications, determine which procedure works best at your installation.

An advantage of using RTTs is you must define fewer DB2 plans in the system. However, when you create new applications, you must update the RTT source to associate new programs with the name of the plan. You use the DB2 DSNMAPN macro to generate RTT entries. After the source is altered, you must recompile and link-edit the RTT.

For IMS batch regions, you can define the DB2 connection in the //DDITV02 file or in an subsystem member (SSM).

For more information on RTTs and how to construct and maintain them, refer to *DB2 Administration Guide*.

Chapter 10. Extracting and Loading Data for IMS-to-DB2 Propagation

This chapter explains the process of extracting data from an IMS database and loading it into a target DB2 table using DataRefresher or your own program. This chapter 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
- LOG-ASYNC asynchronous extract and load considerations

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

Overview of the Extract and Load Process

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

- First, prevent updates to IMS databases using the SCU or the appropriate IMS commands.
- Next, extract and load data into DB2 tables using DataRefresher DEM or a user extract program. Or you can load DB2 rows by running your IMS database load programs in PROP LOAD mode; although this method is not efficient.
- Then, run DB2 utilities such as COPY and RUNSTATS to establish a common point of recovery for IMS and DB2. You might also want to make image copies of the IMS databases.

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:

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

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

Preventing Updates to IMS Databases

If updates are made during the extract and load, data inconsistencies can occur. If you have registered your databases in DBRC, use SCU to prevent updates. The SCU works through the DBRC to prevent or permit updates. If you have not registered your databases in DBRC, you must use alternative methods to prevent the databases from being updated during the extract and load phase. Using the SCU and 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 ACTIVATE to activate propagation requests
2. 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.

For more information on how to register databases in DBRC and use share control, refer to *IMS/ESA Utilities Reference: Database Manager*. For more information on the SCU, see Chapter 13, “Controlling Synchronous Propagation States,” on page 191 and the *IMS DataPropagator Reference*.

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.
- Use the SCU ACTIVATE control statement to activate synchronous propagation for the appropriate propagation requests.
- 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.

Doing the Extract and Load with DataRefresher

Extracting and loading propagated data is simplified if you use DataRefresher. With DataRefresher, mapping and conversions are identical to those done by DPROPNR 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 DPROPNR Map Capture exit (EKYMCE00). When called to extract a propagated, DBRC-registered, full-function database, EKYMCE00 verifies that the database is in read-only status and cannot be concurrently updated by any IMS subsystem. Verification is only possible if DBRC share control is in effect. To perform the validation, EKYMCE00 internally invokes the IMS DBRC utility.
- The DEM provides the extracted and mapped IMS data to the DB2 LOAD utility.

  If Segment or Field exit routines is specified, the DEM calls them during the extract process. The DEM calls are an important part of achieving mapping and conversion identical to those used during propagation.

  The DEM does not call Propagation exit routines during the extract process. The DEM, not your Propagation exit routines, performs mapping and conversion during extract.

Refer to the *Reference* for detailed information on how to code an extract request for DataRefresher.

The extract and load process is illustrated in Figure 32 on page 162.

Extract and Load Process Using DataRefresher

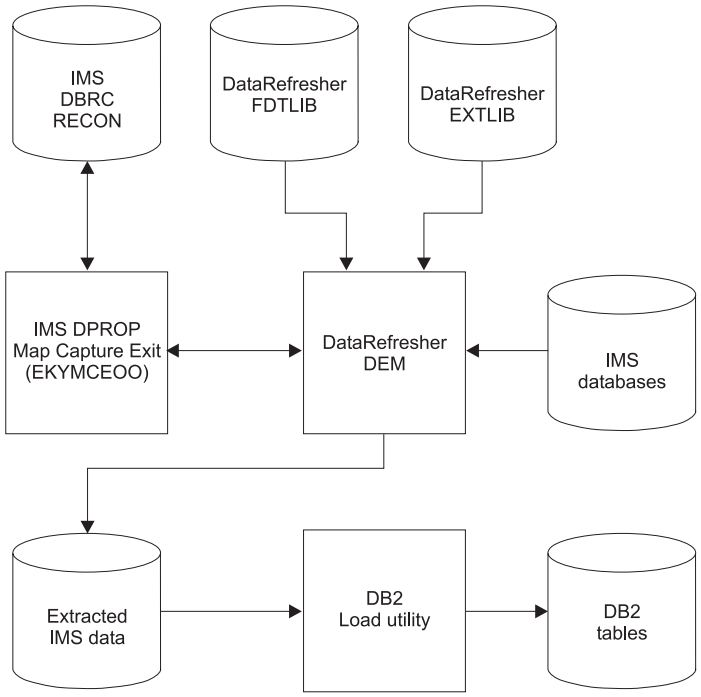


Figure 32. Extract and Load Process Using DataRefresher. After the tables have been loaded, you should run the RUNSTATS utility and copy the tables.

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

- DPROPNR functions are used during the extract process. You must modify the DEM JCL to include the data sets and libraries required by DPROPNR. Refer to the *Reference* for more information on this subject.
- You usually request that the DataRefresher DEM create the control statements for the DB2 LOAD utility by providing a CD keyword on the DataRefresher SUBMIT command. Requesting that the DEM create the load control deck reduces effort and also eliminates one source of potential errors.
- You use the DataRefresher SUBMIT command to request that the DataRefresher DEM create a job to execute the DB2 LOAD utility. Specify the ddname of a file containing skeleton JCL for the DB2 LOAD utility on the JCS keyword of the SUBMIT command.
- For improved performance, you can process DataRefresher extracts of all segments of the same IMS database with a single pass through the database. This practice is called batching DataRefresher extract requests. Batching can save a considerable amount of time and processing. To benefit from extract request batching, you must provide multiple //DXTOUTn DD JCL in the DataRefresher DEM job stream. You must also ensure that all batched extract requests and propagation requests be based on DXTVIEWs that use the same DXTPCB.

  Batching extract requests that belong to generalized mapping cases and extract requests that belong to user mapping cases have restrictions. Refer to the *Reference* for a description of the restrictions.

- If you are extracting and loading data into multiple tables, you might want to run the DB2 LOAD utility jobs in parallel to reduce the amount of elapsed time required to load the tables.

  If the DB2 tables are involved in RIRs, you can:

  – Specify ENFORCE NO on the USERDECK keyword of the DataRefresher SUBMIT command. With ENFORCE NO, the DB2 LOAD utility does not check referential integrity constraints during load processing. When the target table spaces are loaded, DB2 places them in a check-pending state.
  – Run the DB2 CHECK utility after completing all DB2 load jobs.

Doing the Extract and Load with Your Programs

If you do not use DataRefresher to extract data from the IMS database you want to propagate, you must provide programs that extract the IMS data and load the DB2 tables. You can use one of the following methods:

- **Method 1:** Write a program that extracts and maps the IMS data and creates an input file for the DB2 Load utility. Then run the DB2 Load utility to load the data into your DB2 tables. For information on the DB2 Load utility and its input file, refer to the DB2 Command Reference and DB2 Utility Guide and Reference.

  Use this method when you are loading a lot of data into DB2. You have better performance because loading a lot of DB2 data is usually more efficient with the DB2 Load utility than with SQL insert statements.
- **Method 2:** Write a program that extracts and maps IMS data and issues SQL insert statements to insert the data into DB2 tables.

  Using SQL insert statements is usually slower than using the DB2 Load utility. However this method works well for small DB2 tables.
- **Method 3:** Execute your IMS database load programs in PROP LOAD mode. Provide a PROP LOAD control statement in the //EKYIN file allocated to the job step doing the IMS database load. RUP then maps and propagates the IMS inserts to the DB2 tables. With this method, the IMS-to-DB2 mapping, conversion, and propagation is done by RUP.

  RUP uses SQL insert statements to store the data into the DB2 tables. For extension segments of mapping case 2, RUP uses SQL update statements. Using SQL insert statements is usually slower than using the DB2 Load utility. However, this method works well for small DB2 tables.

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

Figure 33 on page 164 is an overview of the three methods of doing the extract and load using your programs.

See the *Reference* for information on how to code an extract request using MVG input tables without DataRefresher.

Extract and Load Process with User-Written Programs

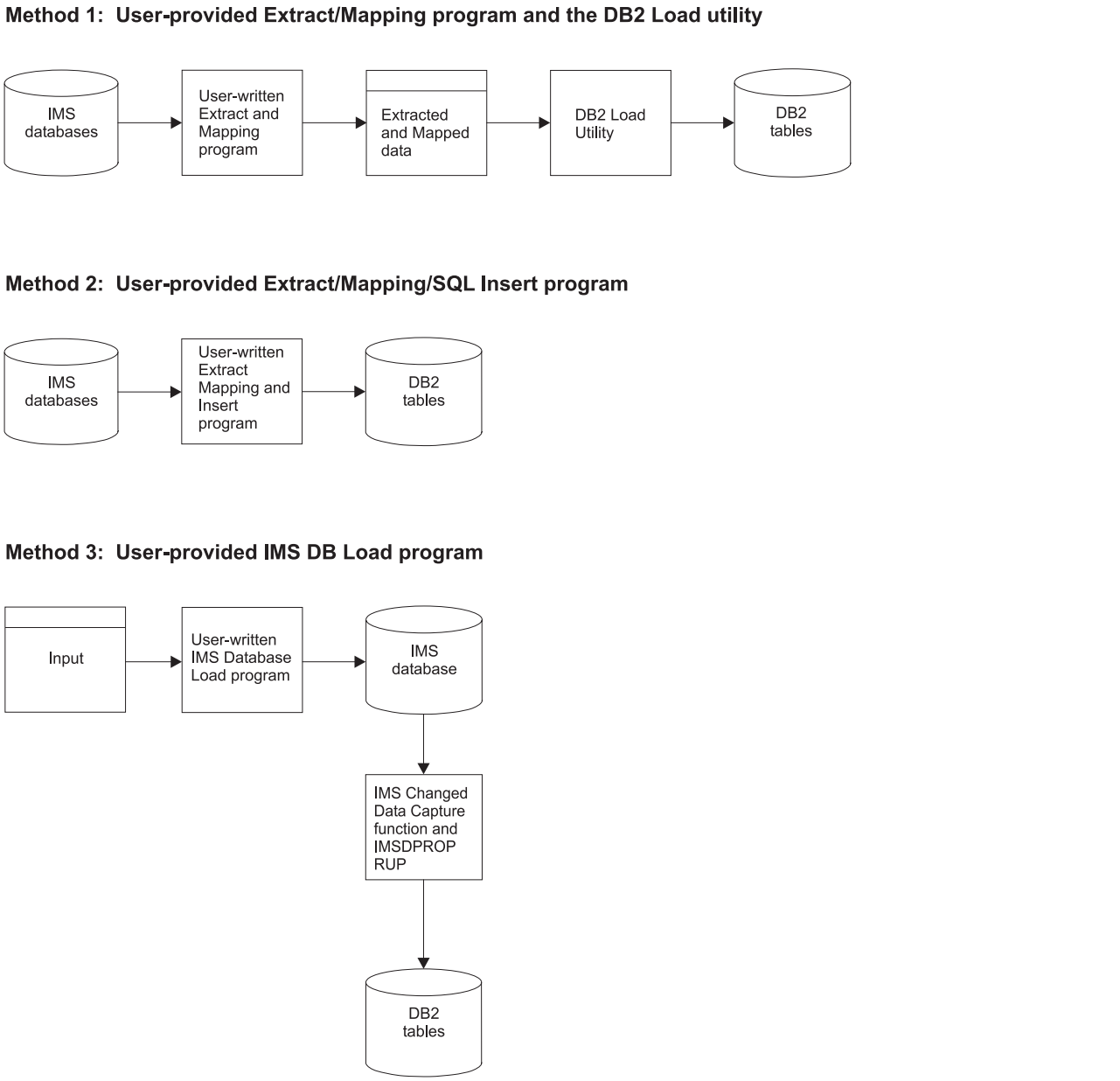


Figure 33. Extract and Load Process with User-Written Programs

Chapter 11. Extracting and Loading Data for DB2-to-IMS (DLU) Propagation

This chapter explains when and how to use the IMS DPROP DL/I Load utilities (DLU). The DLU re-creates or creates an IMS database from the DB2 copy of data propagated by PRTYPE=Es.

Typically, DLU is used only with one-way DB2-to-IMS or two-way synchronous propagation. Using the DLU to re-create a copy of an IMS database implies that the DB2 copy of the data used as input is the master copy. Unless DLU encounters errors, the recreated IMS database is consistent with the DB2 copy of the data.

The DLU can re-create an IMS database in non-error situations, such as when you want to change the mapping definitions or the definitions of the propagated IMS database or DB2 tables.

You can also use DLU as a last resort when the CCU has reported so many discrepancies that you decide the only way to resynchronize IMS and DB2 data is to re-create the IMS database. Some of the reasons why data discrepancies occur are:

- An IMS DPROP emergency stop, using an ESTOP control statement, was issued but the DB2 subsystem had to be kept up for update. For example, propagation might have to be stopped because IMS is down or because some IMS databases or the IMS DPROP directory are unavailable or damaged.
- A point-in-time recovery had to be done for DB2 tables, but no equivalent action could be done for the IMS database.

The following sections consists of:
- An overview
- DLU Restrictions
- DLU Input and Output
- A description of how the DLU selects and processes input data
- Considerations for segments without a unique DL/I key
- Considerations for paired segment types

Overview

In the simplest scenario, all segment types, all segment occurrences, and all fields containing real data are propagated by PRTYPE=Es. In this simple scenario, the only input data for DLU is usually stored in DB2 tables propagated by PRTYPE=Es.

The more common scenario, however, is complex. DLU supports more complex scenarios in which IMS databases contain nonpropagated data. DLU can re-create IMS databases that contain one or more:

- Segment types that are not propagated at all.
- Segment types that are propagated by PRTYPE=U or Ls.
- Occurrences of a propagated segment type that are not propagated. This can happen with propagation requests defined with a WHERE clause or when your Segment exit routines selectively suppress data propagation of some segment occurrences.

- Fields of propagated segments that are not propagated.

To support more complex scenarios, DLU lets you provide the nonpropagated data as additional *complementary input data*. DLU merges the complementary data with the DB2 data propagated by PRTYPE=E.

Complementary data is usually data that is not propagated by PRTYPE=E. If provided, complementary input data is retrieved by DLU from:
- The latest available copy of the IMS database to be recreated. The copy can be either the IMS database itself or an HD unload file.
- One or more user input files.

With the exception of some error scenarios, for example when the DLU needs to discard a child segment occurrence that has no parent, the recreated IMS database are consistent with the propagated DB2 data and also contain any complementary data that is provided. To understand the DLU rules used to merge the propagated DB2 data with the complementary input data, refer to "How the DLU Selects and Processes Input Data" on page 167.

DLU Restrictions

The DLU has the following restrictions:
- The number of DB2 tables from which you can retrieve propagated data is limited to 1024.
- If propagated using PRTYPE=E, segment types without a unique DL/I key field are not necessarily loaded in the sequence expected by your application programs. For more information on this subject, see "Considerations for Segments without a Unique DL/I Key" on page 171.
- After the DLU completes its processing, the position of DEDB subset pointers in the newly created DEDB is not reestablished.
- For physically paired logical child segments, DLU re-creates from DB2 tables *only* that segment type of the pair propagated by a PRTYPE=E. The other segment of the pair cannot be propagated and cannot be recreated by DLU from DB2 data. For that other segment type of the pair, you must give the data to DLU as complementary data. For more information on this subject, see "Considerations for Paired Segment Types" on page 171.

DLU Input and Output

This section describes what the DLU receives as input and generates as output.

DLU Input

The following elements provide input to the DLU:
- Data propagated by a PRTYPE=E stored into the DB2 tables.
- Complementary IMS input data stored in an IMS database or HD unload file. The HD unload file can be created by the IMS HD Unload utility (DFSURGU0) or other database unload utilities, for example, the HSSR DB Unload utilities that create the same HD unload file as DFSURGU0.
- Complementary user input data stored in user input files. The files must be created by a user program before using DLU.

  Complementary user input is especially useful for segments propagated with PRTYPE=U when you do not want to use the current IMS copy to re-create the segments.

Each user input file contains segment data for one IMS segment type. Each record contains segment data, in IMS format, of one segment occurrence, key information, plus some additional information required by DLU. Large IMS segments can be split into multiple consecutive records.

DLU expects that segment occurrences are already sorted in DL/I sequence within user input files. For a description of the expected sort sequence, refer to the *Reference*.

DLU Output

For each execution of the DLU, the output is one IMS database, one HD unload file, or both.

An IMS database recreated by DLU is the same as what would be created by an IMS application program doing an initial database load. If the database has secondary indexes or logical relationships, you must also execute the following IMS utilities before the IMS database is ready to be used by application programs:

- IMS Database Pre-reorganization utility, with a DBIL= control statement
- IMS Database Scan utility, if required by IMS and requested by the IMS Pre-reorganization utility
- IMS Prefix Resolution utility
- IMS Prefix Update utility

How the DLU Selects and Processes Input Data

DLU reads the rows of the DB2 tables propagated by PRTYPE=E. The DLU calls HUP to map the DB2 rows into the IMS segment format. If provided, complementary input data is merged with the propagated data to complete the recreated IMS database. Figure 34 on page 168 shows the role of the DLU in re-creating an IMS database. This process may involve one or more sort operations.

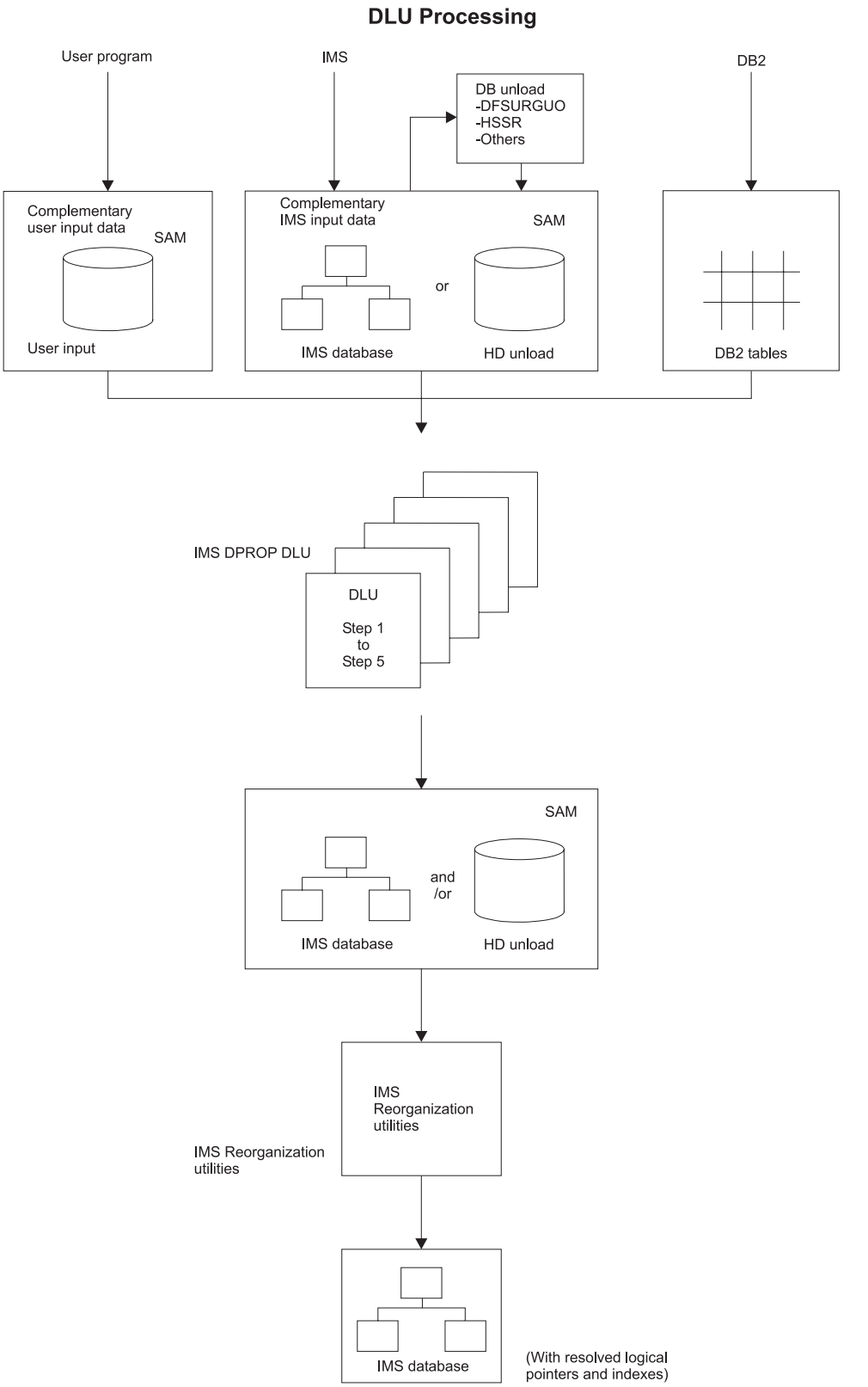


Figure 34. Overview of DLU Processing

As shown in Figure 34, DLU processing consists of five job steps. As explained in *Reference*, you do not always need to execute all 5 job steps.

In the job steps that read the IMS database or DB2 tables, the DLU checks at job step initialization that databases and tables are set to read-only. In addition, to ensuring updates do not occur after job step initialization, DLU issues an SQL statement for each DB2 table for which data is retrieved: LOCK TABLE table name IN SHARE MODE.

The following subsections present scenarios: simple and complex.

Simple Scenario

In the simplest scenario, all segment types, segment occurrences, and fields with real data content in the IMS database are propagated by PRTYPE=Es.

In the simple scenario, you probably re-create or create the IMS database based uniquely on the content of the propagated DB2 tables. The only input data you give DLU are the propagated DB2 tables.

In the unlikely case that you do not want to use the DB2 data as input for some segment types, you identify the segment types on a DLU EXCLUDE statement. For the segment types, you must provide the data as complementary input data, as described in item 1 on page 170.

Complex Scenarios

The scenarios become more complex if some segment types, segment occurrences, or fields are not propagated by PRTYPE=E. You must provide the DLU with complementary input that contains the nonpropagated data in order to include the nonpropagated data in the recreated IMS database. You, therefore, provide DLU with a combination of the following inputs:

- DB2 input, the DB2 tables propagated by PRTYPE=Es
- Complementary input:
  - IMS input data, located in the IMS database or in an HD unload file
  - User input data, located in user input files

  For one specific IMS segment type, DLU retrieves the complementary data either from the complementary IMS input data or from the complementary user input files, never from both. Complementary user input files take precedence over complementary IMS input data.

For each specific IMS segment type, the DLU determines the input sources to be processed. The possible input sources for a particular segment type are:

- Only the DB2 input tables
- Only the complementary data
- Both the DB2 input tables and the complementary data

You might need to understand the rules used by DLU to select and process data from these inputs. The rules differ based on the following segment types:

- Segment types not propagated by PRTYPE=E and segment types excluded with DLU EXCLUDE control statements
- Segment types propagated by PRTYPE=E that can have:
  - Some nonpropagated segment occurrences. These are segment types propagated by PRTYPE=Es defined with a WHERE clause or a Segment exit routine that can suppress propagation.
  - Some nonpropagated fields.

– Neither nonpropagated fields nor nonpropagated segment occurrences. For example, segment types propagated by PRTYPE=E that are defined without a WHERE clause and without a Segment exit routine that can suppress propagation.

The rules for these different segment types are:

1. **For a segment type that is *not* propagated by PRTYPE=E** and for segment types that have been excluded with a DLU EXCLUDE control statement:

   The only input processed by DLU is the complementary input. DLU loads one segment occurrence for every occurrence found in the complementary input data.

2. **For a segment type propagated by PRTYPE=E that can have some nonpropagated segment occurrences** if the segment type is not excluded by a DLU EXCLUDE control statement:

   The DLU processes both the DB2 input and the complementary input. DLU loads into the recreated IMS database one segment occurrence:

   - For each segment occurrence mapped from the DB2 input tables
   - For each segment occurrence located in the complementary data that should *not* be mapped to DB2. Occurrences not mapped are based on WHERE clauses and Segment exit routines of your propagating PRTYPE=E.

   To prevent inconsistencies, between the recreated IMS database and the DB2 tables, DLU does *not* load into the recreated IMS database a segment occurrence found in the complementary data if both:

   - The segment occurrence *is* supposedly mapped to DB2, based on the WHERE clauses and Segment exit routines
   - DLU does not find a matching DB2 row in the DB2 input tables

3. **For a segment type propagated by PRTYPE=E that has some nonpropagated fields** if the segment type is not excluded by a DLU EXCLUDE control statement:

   DLU processes both the DB2 input and, if provided, the complementary input. DLU tries to match every segment occurrence mapped from the DB2 input rows with segment occurrences located in the complementary input data.

   - If a match is found in the complementary input data, nonpropagated fields are filled with the content of the complementary data.
   - If a match is not found, nonpropagated fields are filled with a default value (zeroes or blanks).

4. **For a segment type propagated by PRTYPE=E that can have neither nonpropagated fields nor nonpropagated occurrences** if the segment type is not excluded by a DLU EXCLUDE control statement:

   The only input processed by DLU is the DB2 input. DLU loads one segment occurrence for every segment occurrence mapped from the DB2 input tables.

   DLU ignores complementary input for this segment type. Even if you provide complementary input, DLU does *not* load into the recreated IMS database a segment occurrence found in the complementary data if DLU does not find a matching DB2 row in the DB2 input. By ignoring the complementary input, DLU avoids creating inconsistencies between the recreated IMS database and the DB2 tables.

Considerations for Segments without a Unique DL/I Key

If your IMS database contains segment types without a unique DL/I key field, DLU might load the segments in a sequence different from the sequence expected by your IMS application programs. You might need to understand in which sequence the segment types are loaded by DLU.

- **For segment types not propagated by PRTYPE=E**, DLU loads the segments in the sequence of the complementary input. If the complementary input is an IMS database or an HD unload file, the sequence is the same as the previous copy of the IMS database. If the complementary input is a user file, the sequence is the same as the input segments within the user input file.
- **For segment types propagated by PRTYPE=E**, even if they are excluded through a DLU EXCLUDE control statement:

  When loading dependent segments under their physical parent, DLU sorts the dependents in the following sequence:

  1. In EBCDIC ascending sequence of the DL/I key field, if a DL/I key field exists.
  2. In EBCDIC ascending sequence of those non-key DL/I fields that are mapped to a DB2 primary key column. If a segment has more than one such non-key DL/I fields, the DLU sort criteria include these fields in the ascending order of their starting position within the segment.

Considerations for Paired Segment Types

With paired logical child segment types propagated with a PRTYPE=E, you must consider:

- Physically paired segment types
- Virtually paired segment types

Physically Paired Segment Types

If you have IMS logical relationships with physically paired logical children, only one of the two segment types of the pair can be propagated with IMS DPROP. DLU supports the propagated logical child the same way it supports any other propagated segment type. If propagated by PRTYPE=E, the segment type is created by DLU based on input in the propagated DB2 tables.

DLU does not provide any specific support for the *other*, nonpropagated segment type of the pair. Including the occurrences of the other segment type into the recreated IMS database is your responsibility. You must:

- Provide the occurrences of the nonpropagated segment type in the complementary input data
- Ensure that the occurrences of both segment types of the pair are consistent

The following types of pairing can occur:

- Paired segment types within the same IMS database
- Paired segment types across two IMS databases

Within the Same IMS Database

When you are propagating paired segment types within the same IMS database, the following points apply:

- When none of the paired segment types is propagated with PRTYPE=E, you must provide data for both segment types in complementary data. And you must ensure that the segment occurrences of both types are consistent.

- When one of the paired segment types is propagated with PRTYPE=E, the DLU creates the data of the propagated segment type from the DB2 rows. DLU does not provide any special processing for the other paired segment type.

  Recommendation: The data for the nonpropagated segment type should be provided in complementary data. Before running the DLU, you should run a user-written program to ensure data is consistent for the paired data.

Across Two IMS Databases

When you are propagating paired segment types across two IMS databases, the following points apply:

- When neither of the paired segment types is propagated with PRTYPE=E and you re-create one of the two IMS databases, you must use complementary input to provide the data for that segment type contained in the recreated database. Before running the DLU, you should run a user-written program to ensure data consistency for the paired data across the two IMS databases.
- When the IMS database to be recreated contains a paired segment type propagated with a PRTYPE=E, the DLU treats that segment type the same as it would any other propagated segment type; DLU creates the segment occurrences from the DB2 input rows. However, before executing the DLU, you should run a user-written program to ensure data is consistent for the paired data across the two IMS databases.
- When the IMS database to be recreated contains the paired segment type that is not propagated, the DLU treats that segment type the same as it would any other nonpropagated segment type. You must provide its data in complementary data. Before executing the DLU, you should run a user-written program to ensure data is consistent for the paired data across the two IMS databases.

Virtually Paired Segment Types

The physically existing segment type of a pair is supported by DLU without restrictions.

As expected by IMS, DLU does not load the virtual segment type of a pair. Therefore, if you are providing complementary user input files, you do not need to provide them for the virtual segment types.

Part 4. Propagating Data with IMS DPROP

Chapter 12. Performing Synchronous Propagation 175
Normal RUP Processing 175
Environment 175
Processing 175
Generalized Mapping Case 176
User Mapping Case 176
Normal HUP Processing. 176
HUP Environment. 176
HUP Processing 176
Generalized Mapping Case 177
User Mapping Case 177
Error Handling Options 177
Dynamic Backout in IMS Environments . . . 178
DB2 Region Error Option 178
IMS DPROP Error Option 178
ERROPT=BACKOUT 178
ERROPT=IGNORE 179
IMS INIT STATUS Call 179
IMS Support for the INIT STATUS GROUPA Call 180
IMS Support for the INIT STATUS GROUPB Call 180
RUP and HUP Support for the INIT STATUS GROUPA Call 180
RUP and HUP Support for the INIT STATUS GROUPB Call 181
Usage Notes. 182
Use of MODE=SNGL. 183
RUP and HUP Error Processing 183
Severe Errors 185
DB2 Deadlocks 185
IMS Deadlocks 185
Propagation Emergency Stopped or Deactivated 185
Unavailable Resources 186
Other Errors. 186
Summary of Error Handling 186
Some Causes of Unavailable Resources 187
RUP and HUP Error Reporting 188
Limiting the Number of Error Messages Resulting From ERROPT=IGNORE 188
Using MVS to Suppress Messages 189

Chapter 13. Controlling Synchronous Propagation States 191
Synchronous Propagation States and Modes . . . 191
Synchronous Propagation State of the Entire IMS DPROP System 191
Synchronous Propagation Status of Individual Propagation Requests. 191
PROP OFF Mode for DB Repair Programs. . . 192
Read-Only Status of IMS Databases 193
Read-Only Access Mode of DB2 Table Spaces and Databases 194
Status Change Utility (SCU) 194
Controlling Propagation Requests 195
Changing the Status of Propagation Requests Groups 195
Making Orderly Status Changes 196
Activating Propagation Requests 197
Deactivating and Emergency Deactivating Propagation Requests. 198
Suspending Propagation Requests 199
Controlling Full-Function IMS Databases . . . 200
READON 200
READOFF 200
Controlling DB2 Databases and Table Spaces 201
READON 201
READOFF 201
Controlling the IMS DPROP System 201
ESTOP 202
RESET. 202
General Service Functions of the SCU 202
Turning Synchronous Propagation Off Using ALLOWPROPOFF and DENYPROPOFF . . 203
Displaying System Information using DISPLAY, LIST.DB, -DISPLAY DATABASE. . 203
Changing Error Options Using ERROPT . . 204
Changing Error Control Information Using ERRCTL 204
Initializing the IMS DPROP System, Status File, and VLF Objects (INIT) 204
Turning Tracing On and Off (TRACEON, TRACEOFF). 205
RUP and HUP Control Statements 205
Controlling Synchronous Propagation Using PROP Control Statements 206
PROP LOAD 206
PROP OFF 206
PROP SUSP 206
Relationship of PR Status and PROP SUSP/OFF Control Statements 207
Controlling Traces 207
TRACE 207
TRDEST 208
Controlling the Number of Resident SQL Update Modules and PRCBs 208
Resident SQL Update Modules 208
Resident PRCBs 209

Chapter 14. Database Maintenance for Synchronous Propagation 211
Checkpoint and Restart in the IMS and DB2 Environment 211
Restart of IMS Online and DB2 212
Checkpoint and Restart of an IMS Batch Program 212
Database Backout for IMS Batch Programs . . . 212
IMS Dynamic Backout for Batch Regions . . . 212
Backout of Committed Data 212
Backup and Recovery 213
System Data Sets 213

Databases 213
Timestamp Recovery 214
Data Resynchronization 214
Database Repair 215
IMS and DB2 Repair Functions 215
User-Written Repair Programs 215
Preventing Inadvertent Execution of Repair Programs 216
Database Reorganization and Load 216
Initial Load of IMS Databases 217
Load of DB2 Tables 217
CCU Verification 217
IMS DPROP Directory Recovery 217

Chapter 15. Verifying Data Consistency (CCU) 219
Overview of the CCU 219
When to Use the CCU 220
CCU Considerations for Synchronous Propagation 221
Considerations When Concurrent Updates Are Being Done 221
Data Availability 221
DB2 Referential Integrity Constraints 221
Running the CCU 222
Phases of the CCU 222
CCU Verification Techniques 222
Direct Technique 222
Hashing Technique 223
Types of Inconsistencies and Generated Repair Statements 223
Generated SQL Repair Statements 224
Generated DL/I Repair Statements 224
Large Numbers of Inconsistencies 224
Some Reasons for Inconsistencies 224

Chapter 16. IMS DPROP's Problem Determination Tools 227
IMS DPROP Trace Facilities 227
IMS DPROP Audit Facilities 228
Using SMF 228
Audit Extract Utility and Audit Trail Table . . 228
Creating an Audit Trail 229
Audit Trail Table Security 230
Comparison of Audit and Trace Information . . 230
CCU and the Audit Trail 230
Monitoring Consistency with the CCU 231
Monitoring Propagation with the Message Table of the IMS DPROP Directory 231

Chapter 17. IMS DPROP Performance and Monitoring 233
IMS DPROP Performance 233
Mapping and Design Phase 233
Setup Phase 234
IMS-to-DB2 Propagation 234
DB2-to-IMS Synchronous Propagation . . . 234
Propagation Phase: Synchronous Propagation Performance 234
IMS-to-DB2 Synchronous Propagation . . . 234
DB2-to-IMS Synchronous Propagation . . . 235
Two-Way Synchronous Propagation 236
Propagation Phase: User Asynchronous Propagation Performance 237
CCU Execution 237
Monitoring Propagation 238

Part 4 covers the propagation phase and maintenance and control phase of data propagation. Part 4 consists of these chapters:

- Chapter 12, "Performing Synchronous Propagation," on page 175, describes normal and error processing in a synchronous environment.
- Chapter 13, "Controlling Synchronous Propagation States," on page 191, describes propagation states and how to control them using the IMS DPROP Status Change utility (SCU) and IMS DPROP RUP/HUP control statements.
- Chapter 14, "Database Maintenance for Synchronous Propagation," on page 211, describes various aspects of database maintenance, such as checkpoint and restart, backout, backup and recovery, etc.
- Chapter 15, "Verifying Data Consistency (CCU)," on page 219, describes how to verify whether your IMS and DB2 data are consistent. This is done using the Consistency Check utility (CCU).
- Chapter 16, "IMS DPROP's Problem Determination Tools," on page 227, 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 17, "IMS DPROP Performance and Monitoring," on page 233, discusses how to optimize performance in the IMS DPROP environment and monitor propagation using IMS and DB2 tools.

Chapter 12. Performing Synchronous Propagation

This chapter describes:
- Normal processing during propagation
- Options for handling errors during propagation
- Error processing
- Error reporting and suppression of error messages

for synchronous propagation.

Normal RUP Processing

This section briefly describes the environment and processing sequence of the RUP during synchronous propagation.

Environment

RUP runs as an IMS DPROP-supplied IMS Data Capture exit. You specify its use by coding the EXIT=EKYRUP00 keyword on the IMS DBD.

When a propagating IMS application updates a database, IMS calls RUP in the application's address space just before returning from the update. RUP runs in the same environment as the propagating application.

For updating calls, IMS calls RUP each time a segment is changed and must be propagated. When IMS DPROP is called, IMS passes data to RUP based on the options specified on the EXIT= keyword of the DBD. If you use default values, IMS passes the following data to the RUP:
- Physical DBD name and physical segment name
- Fully concatenated key of the segment
- Segment data

Refer to "Creating or Changing DBDs" on page 105 for additional information on data that can be passed by IMS.

Processing

When RUP receives the changed data segment, its processing is based on the current state of the IMS DPROP system and the propagation request status for the segment type that was updated. If the system is emergency stopped, RUP returns without propagating anything. Otherwise, RUP gets the propagation request status from the RUP PRCB. The RUP PRCB can contain one or more propagation requests. If there are multiple propagation requests, RUP processes them sequentially.

Based on propagation request status and the control statements in the //EKYIN data set of the updating batch or dependent region, RUP determines whether to allow the update and whether to propagate it. RUP follows these rules:

1. If you have provided a PROP OFF control statement in the //EKYIN data set, RUP does not propagate the update. You must have allowed PROP OFF previously, using the SCU.

2. If the propagation request is active, RUP propagates the update. If the propagation request is active but there is a PROP SUSP control statement in the //EKYIN data set, updates are not allowed.
3. If the propagation request is suspended, RUP does not propagate the update. Only updates of programs with PROP SUSP control statements are allowed.
4. If the propagation request is inactive, RUP does not propagate the update.

See Chapter 13, "Controlling Synchronous Propagation States," on page 191 for more information about propagation request status and the //EKYIN data set.

If the update is to be propagated, RUP uses the mapping information in the propagation request for the changed segment. The propagation request tells RUP whether to propagate the update using a generalized or user mapping case.

Generalized Mapping Case

If RUP does the propagation with a generalized mapping case, the RUP:

1. Calls the Segment and Field exit routines, if applicable. If the IMS format of the data cannot be propagated to the DB2 table by RUP alone, the exit routines can convert the data into a IMS DPROP format that RUP can then process.
2. Calls its data conversion routines. These routines support a number of standard data conversions. If RUP detects that the type of data in its IMS DPROP format is different from the propagated DB2 column, it converts the data before propagation.
3. Then provides DB2 column values to the SQL update module for the current propagation request; the SQL update module then applies the updates to the DB2 table.

User Mapping Case

If the update is to be propagated with a user mapping case, RUP calls the user-written Propagation exit routine. RUP also provides support services such as tracing, auditing, and error handling.

Normal HUP Processing

This section briefly describes the environment and processing of HUP.

HUP Environment

HUP runs as the IMS DPROP-supplied DB2 Data Capture exit. HUP has the alias name of DB2CDCEX. You specify HUP's use by coding the DATA CAPTURE CHANGES option on the CREATE TABLE or ALTER TABLE statement.

When a propagating IMS application updates a DB2 table, DB2, after executing the SQL statement, tells IMS to invoke HUP in the application's address space. HUP runs in the same environment as the propagating application.

The rows that are to be propagated by IMS DPROP are retrieved by HUP using DB2 IFI calls. For SQL statements affecting multiple rows, DB2 returns to the HUP the data of all changed rows.

HUP Processing

When HUP receives a changed data row, its processing is based on the current state of the IMS DPROP system and the status of the propagation request for the

table that was updated. If IMS DPROP is emergency stopped, HUP returns without propagating anything. Otherwise, HUP gets the propagation request status from the HUP PRCB.

Based on propagation request status and the control statements in the //EKYIN data set for the updating batch or dependent region, HUP determines whether to allow the update and whether to propagate it. HUP follows these rules:

1. If you have provided a PROP OFF control statement in the //EKYIN data set, HUP does not propagate the update. You must have allowed PROP OFF previously, using the SCU.
2. If the propagation request is active, HUP propagates the update. If the propagation request is active but there is a PROP SUSP control statement in the //EKYIN data set, updates are not allowed.
3. If the propagation request is suspended, HUP does not propagate the update. Only updates of programs with PROP SUSP control statements are allowed.
4. If the propagation request is inactive, HUP does not propagate the update.

See Chapter 13, "Controlling Synchronous Propagation States," on page 191 for more information about propagation request status and the //EKYIN data set.

If the update is to be propagated, HUP uses the mapping information in the propagation request for the changed table. The propagation request tells HUP whether to propagate the update using a user or a generalized mapping case.

Generalized Mapping Case

If HUP does propagation with a generalized mapping case, the HUP:

1. Calls its data conversion routines. The routines support a number of standard data conversions. If HUP detects that the type of data in its DB2 format is different from the DataRefresher and IMS DPROP format, it converts it.
2. Calls the Field exit routine, if applicable. If the DataRefresher and IMS DPROP format of the field cannot be propagated to the IMS segment by HUP alone, the Field exit routine can convert the data to an IMS format.
3. Depending on the type of update, retrieves the segment from the IMS database.
4. HUP calls the Segment exit routine, if applicable. The Segment exit routine maps an IMS segment from its format in the propagation request definition to its real format in the IMS database. Key fields and ID fields cannot be changed by the Segment exit routine.
5. Then applies the update to the IMS database.

User Mapping Case

If the update is to be propagated with a user mapping case, HUP calls the user-written Propagation exit routine. HUP also provides support services such as tracing, auditing, and error handling.

Error Handling Options

This section discusses error handling as it relates to IMS DPROP operations. In synchronous mode, RUP and HUP handle errors when propagation fails. Topics in this section include:

- Dynamic backout in IMS environments
- DB2 region error option
- IMS DPROP error option
- IMS INIT STATUS call

Dynamic Backout in IMS Environments

When propagation fails, you must back out the changes made to IMS databases so that IMS and DB2 databases remain consistent. For online environments, IMS does the backout automatically; you do not need to take any special action.

In an IMS batch environment, we recommend use of IMS dynamic backout. To use dynamic backout:

- Allocate the IMS batch log to a disk data set (//IEFRDER DD statement of the DLIBATCH or DBBBATCH JCL procedures)
- Specify a BKO=Y keyword on the DLIBATCH or DBBBATCH JCL procedure

IMS dynamically backs out database updates made by batch applications when propagation fails. Otherwise, you must run the IMS Batch Backout utility (DFSBBO00) to back out the database changes. If DB2 deadlock occurs, you cannot do a dynamic backout of the IMS database. If deadlock occurs during propagation, you must run the Batch Backout utility, even if you follow the recommendations for dynamic backout.

For more information on running the IMS Batch Backout utility, see *IMS/ESA Utilities Reference: Database Manager*. For information on the keywords used by IMS JCL procedures, refer to *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

DB2 Region Error Option

The DB2 Region Error Option (REO) determines if control is to be returned to an IMS application if DB2 or some DB2 resources are unavailable. The REO option has no effect if either the IMS DPROP directory or the propagated tables are unavailable.

For IMS DPROP, an REO of R is recommended. With this option, RUP and HUP can do their usual error handling if DB2 is not available. With other REO options, the application is abended.

You can specify the DB2 REO in the SSM member of IMS PROCLIB. For batch regions, if your JCL includes a //DDITV02 DD statement, specify the REO option in the control statement of the //DDITV02 file.

IMS DPROP Error Option

When you define a propagation request, you also specify the error option (ERROPT). ERROPT determines what RUP and HUP are to do when propagation fails. Specify ERROPT as either:

- BACKOUT
- IGNORE

ERROPT is recorded in the propagation request table in the IMS DPROP directory. You can change ERROPT with the SCU ERROPT control statement without regenerating the propagation request. For more information on the SCU and ERROPT, refer to the *Reference*.

ERROPT=BACKOUT

If you specify BACKOUT and if propagation fails, RUP and HUP back out all database changes made since the last commit point or point of consistency. By backing out changes when a failure occurs, you maintain consistency between IMS

and DB2 data. As explained in "RUP and HUP Error Processing" on page 183, backout is handled slightly differently depending on the type of error situation.

The use of ERROPT=BACKOUT can affect availability of IMS applications when propagation fails and backout is underway. However, BACKOUT is recommended in most cases for maintaining consistency.

ERROPT=IGNORE

If you specify IGNORE, RUP and HUP ignore propagation failures except when they occur due to unavailable resources and deadlocks. No failure indicators are returned to the application program, but IMS DPROP does write error messages.

If the failure is due to unavailable resources or deadlocks, RUP and HUP back out all database changes since the last commit point. IMS DPROP tracks these specific errors to ensure the errors are temporary and that data consistency is maintained. Deadlocks or short periods of unavailability, if ignored, can cause increasingly inconsistent IMS or DB2 data. See "Severe Errors" on page 185, "DB2 Deadlocks" on page 185, and "IMS Deadlocks" on page 185 for a discussion of deadlocks and severe errors.

While use of ERROPT=IGNORE reduces the number of propagation failures that can impact availability of IMS applications, it can result in inconsistencies between IMS and DB2 data.

Using ERROPT=IGNORE is helpful when you are beginning to implement propagation and are not yet sure whether your mapping and conversions are correct. After your propagation definitions are validated, we recommend that you change to ERROPT=BACKOUT.

IMS INIT STATUS Call

A propagating IMS application can issue an:

- IMS INIT STATUS GROUPA call to regain control when:
  - Your application program's IMS calls try to access unavailable IMS data
  - IMS DPROP encounters unavailable resources or propagation failures
- IMS INIT STATUS GROUPB call to regain control when one of the situations for the GROUPA call occurs and additionally when:
  - An IMS deadlock is encountered while processing your application program's IMS calls
  - An IMS or DB2 deadlock is encountered while processing IMS or DB2 calls issued by IMS DPROP, but only in a non-message driven BMP environment

Issuing an INIT STATUS GROUPx call lets applications do some alternate processing when unavailable data, deadlocks or propagation failures occur.

The IMS and IMS DPROP support for INIT STATUS GROUPx calls is similar. The following sections describe:

- IMS support for the INIT STATUS GROUPA call
- IMS support for the INIT STATUS GROUPB call
- RUP and HUP support for the INIT STATUS GROUPA call
- RUP and HUP support for the INIT STATUS GROUPB call
- Usage notes for INIT STATUS call
- Use of MODE=SNGL

IMS Support for the INIT STATUS GROUPA Call

An IMS application signals its sensitivity to data availability by issuing INIT STATUS GROUPA. Issuing this call tells IMS that the application is to regain control after an attempt to access unavailable data is made. Use of the call implies that the application has logic to deal with this situation.

- If an IMS application does *not* issue an INIT STATUS GROUPA call, it is insensitive to data availability. IMS aborts the application if it tries to access unavailable data. IMS backs out any update of the failing UOW, cancels any non-express output messages, and puts any input message the program was processing in the suspend queue for eventual reprocessing.
- If the IMS application issues an INIT STATUS GROUPA call, it is sensitive to data availability. IMS returns either a BA or BB status code to the application if it tries to access unavailable data. Before returning a BB status code, IMS backs out all database updates made by the application since its last commit point. Also, all non-express messages sent by the application since the last commit point are cancelled. The position of database PCBs is set to the beginning of the database.

For information on how to issue the IMS INIT call and on IMS status codes, see *IMS/ESA Application Programming: DL/I Calls*.

IMS Support for the INIT STATUS GROUPB Call

IMS support for the INIT STATUS GROUPB call is similar to that for the GROUPA call. With GROUPB, the application signals to IMS that, in addition to the conditions of the GROUPA call, it is also sensitive to deadlocks. Use of the call implies that the application has logic to deal with this situation.

For more information on INIT STATUS GROUPB, refer to *IMS/ESA Application Programming: DL/I Calls*.

RUP and HUP Support for the INIT STATUS GROUPA Call

IMS DPROP support for the INIT STATUS GROUPA call is similar to the IMS support. When propagation fails, IMS DPROP determines if the propagating IMS application has issued an INIT STATUS GROUPA call.

- If the application *has not* issued an INIT STATUS GROUPA call, the application does not get control when propagation fails (exceptions are explained in "ERROPT=IGNORE" on page 179). When propagation fails, RUP and HUP initiate a backout of the failing UOW.
  - For online regions, if failure is due to unavailable resources, RUP and HUP initiate backout by issuing an IMS ROLS call. ROLS backs out any update of the failing UOW, cancels any non-express output messages, and puts any input message the program was processing in the suspend queue for eventual reprocessing. The ROLS call results in an IMS pseudo-abend.
  - For batch regions or if the failure is not due to unavailable resources, RUP and HUP initiate the backout by issuing an IMS ROLB call and an abend.
- If the application *has* issued an INIT STATUS GROUPA call, the application gets control when propagation fails (exceptions are deadlocks and severe errors).

  When propagation fails, RUP and HUP issue an IMS ROLB call to initiate a backout of the failing UOW. They return either a BB status code (RUP) or -929 SQL error code and '58002' SQLSTATE (HUP) to the application. IMS DPROP identifies one of two reasons for the failure when returning an error to an application sensitive to unavailable resources. RUP gives the reason in the segment name field of the PCB; HUP gives the reason in the SQLERRMC field of the SQLCA, as described:
  - Unavailable resource failure (PROPUNAV)

If a propagation request fails because of an unavailable resource, then RUP returns a BB status code and the string PROPUNAV in the segment name field of the PCB. HUP returns a -929 SQL error code ('58002' SQLSTATE) and the string PROPUNAV in the first 8 bytes of the SQLERRMC field of the SQLCA.

– Other propagation failure (PROPOTHR)

  Examples of other failures include non-numeric data in an IMS field defined as numeric, SQL"not found" conditions encountered when trying to replace a DB2 row, or IMS"not found" conditions encountered when trying to replace an IMS segment. For these failures, RUP returns a BB status code and the string PROPOTHR in the segment name field of the PCB. HUP returns a -929 SQL error code ('58002' SQLSTATE) and the string PROPOTHR in the first 8 bytes of the SQLERRMC field of the SQLCA.

The ROLB call issued by RUP and HUP resets the position of all database PCBs to the beginning of the database and closes all open SQL cursors, regardless of whether they were defined as WITH HOLD.

For a more detailed description of RUP and HUP's error handling logic, refer to "RUP and HUP Error Processing" on page 183. For a summary of actions performed in error situations, refer to "Summary of Error Handling" on page 186.

RUP and HUP Support for the INIT STATUS GROUPB Call

A propagating IMS application can issue an IMS INIT STATUS GROUPB call to regain control when:

- IMS application program calls try to access unavailable IMS data
- IMS DPROP encounters unavailable resources or propagation failures
- IMS application program calls encounter an IMS deadlock
- IMS DPROP encounters an IMS or DB2 deadlock in a non-message-driven BMP environment

For deadlock situations, RUP and HUP support of INIT STATUS GROUPB differs depending on the IMS environment. When encountering deadlock, IMS DPROP determines if the propagating IMS application has issued an INIT STATUS GROUPB call.

- If the application *has not* issued an INIT STATUS GROUPB call or if the environment is either a message-driven MPP/BMP or pure batch, then the application does not get control when deadlock occurs. When deadlock occurs, backout of the failing UOW is initiated.
  – For online regions, if failure is due to unavailable resources, RUP and HUP initiate backout by issuing an IMS ROLS call. ROLS backs out any update of the failing UOW, cancels any non-express output messages, and puts any input message the program was processing on the input queue for eventual reprocessing. And the ROLS call results in an IMS pseudo-abend.
  – For batch regions, RUP and HUP initiate backout and then abend the region. In case of a DB2 deadlock in a pure batch environment, the IMS database is *not* backed out. You need a batch backout to undo the changes to the IMS database.
- If the application *has* issued an INIT STATUS GROUPB call and is running in a non-message driven BMP environment, the application gets control if deadlock occurs.

  When detecting deadlock, RUP and HUP return either a BC status code (RUP) or a -911 SQL error code and '40000' SQLSTATE (HUP) to the application.

Additionally, RUP returns the string PROPDLOK in the segment name field of the PCB, and HUP returns the string PROPDLOK as first token in the SQLERRMC field of the SQLCA.

The ROLB call issued by the database manager resets the position of all database PCBs to the beginning of the database and closes open SQL cursors, even if the cursors have been defined with the WITH HOLD option.

For a more detailed description of RUP and HUP's error handling logic, refer to "RUP and HUP Error Processing" on page 183. For a summary of actions performed in error situations, refer to "Summary of Error Handling" on page 186.

Usage Notes

- A propagating application issuing INIT STATUS GROUPA or GROUPB should be prepared to receive a:
  - BB status code after any propagating IMS call
  - -929 SQL error code ('58002' SQLSTATE) after any propagating SQL call
- A propagating application running in a non-message driven BMP environment and issuing INIT STATUS GROUPB should additionally be prepared to receive a:
  - BC status code after any propagating IMS call
  - -911 SQL error code ('40000' SQLSTATE) after any propagating SQL call
- RUP does not return a BA status code. An application that issues INIT STATUS GROUPA should be prepared to deal with BA and BB status codes. BA could be returned when IMS detects that the IMS data required to process a call is unavailable.
- RUP does not return an FD status code. An application running in a non-message driven BMP environment and issuing INIT STATUS GROUPB should be prepared to deal with BA, BB, BC, and FD status codes. FD could be returned when IMS detects a deadlock for a DEDB or MSDB resource outside of a propagating call.

If propagating transaction codes are defined to IMS as single mode transactions (MODE=SNGL) then:

- If RUP returns a BB status code or HUP returns a -929 SQL error code ('58002' SQLSTATE) to a message processing program, the following events occur after input messages are retrieved by the program:
  - If the application issues a GN call to the I/O PCB after the BB status code is received, IMS returns a QD status code
  - If the application issues a GU call to the I/O PCB after the BB status code or -929 SQL error code ('58002' SQLSTATE) is received, IMS returns the first segment of the *next* message or a QC status code if there are no more messages for the program.
- When encountering a BB status code or -929 SQL error code ('58002' SQLSTATE), a message-driven program can insert a response containing an error message into the I/O PCB. The response can indicate to the terminal user that resources required to process the transaction are temporarily unavailable.
- When encountering a BB status code or -929 SQL error code ('58002' SQLSTATE), a message-driven program can save the failing input message to be reprocessed later when the required data becomes available. To do so, the application can issue a ROLS call against the I/O PCB without a token, resulting in a U3303 abend; the input message is placed on the IMS suspend queue.

Use of MODE=SNGL

For online IMS applications issuing INIT STATUS GROUPA or GROUPB calls, the use of MODE=SNGL (single) is recommended rather than MODE=MULT. MODE is specified in the TRANSACT macro for transactions defined to IMS.

With MODE=MULT, when RUP or HUP issues the ROLB call, IMS behaves as if the application had again retrieved all message segments of the *first* transaction of the failing UOW. IMS assumes the application is reprocessing the first transaction. The transaction might not be the one that caused propagation to fail. Generally, the application no longer has the first message of the UOW available and cannot perform any reasonable processing for the first transaction.

RUP and HUP Error Processing

This section describes how RUP and HUP handle various types of propagation failures for propagation requests for either the generalized or user mapping cases.

In general, if ERROPT=IGNORE is specified, IMS DPROP ignores propagation failures except for those caused by unavailable resources and deadlocks. If ERROPT=BACKOUT is specified, RUP and HUP back out changes made since the last commit point. RUP and HUP back out changes differently for different types of errors.

Figure 35 on page 184 summarizes RUP and HUP error processing. The figure shows error processing from the application program's perspective, describing the actions taken by a combination of RUP or HUP, IMS, and DB2.

For additional information on diagnostic messages and traces when errors occur, see *Diagnosis*. Topics in this section are:

- Severe errors
- DB2 deadlocks
- IMS deadlocks
- Propagation emergency stopped or deactivated
- Unavailable resources
- Other errors
- Summary of error handling
- Some causes of unavailable resources

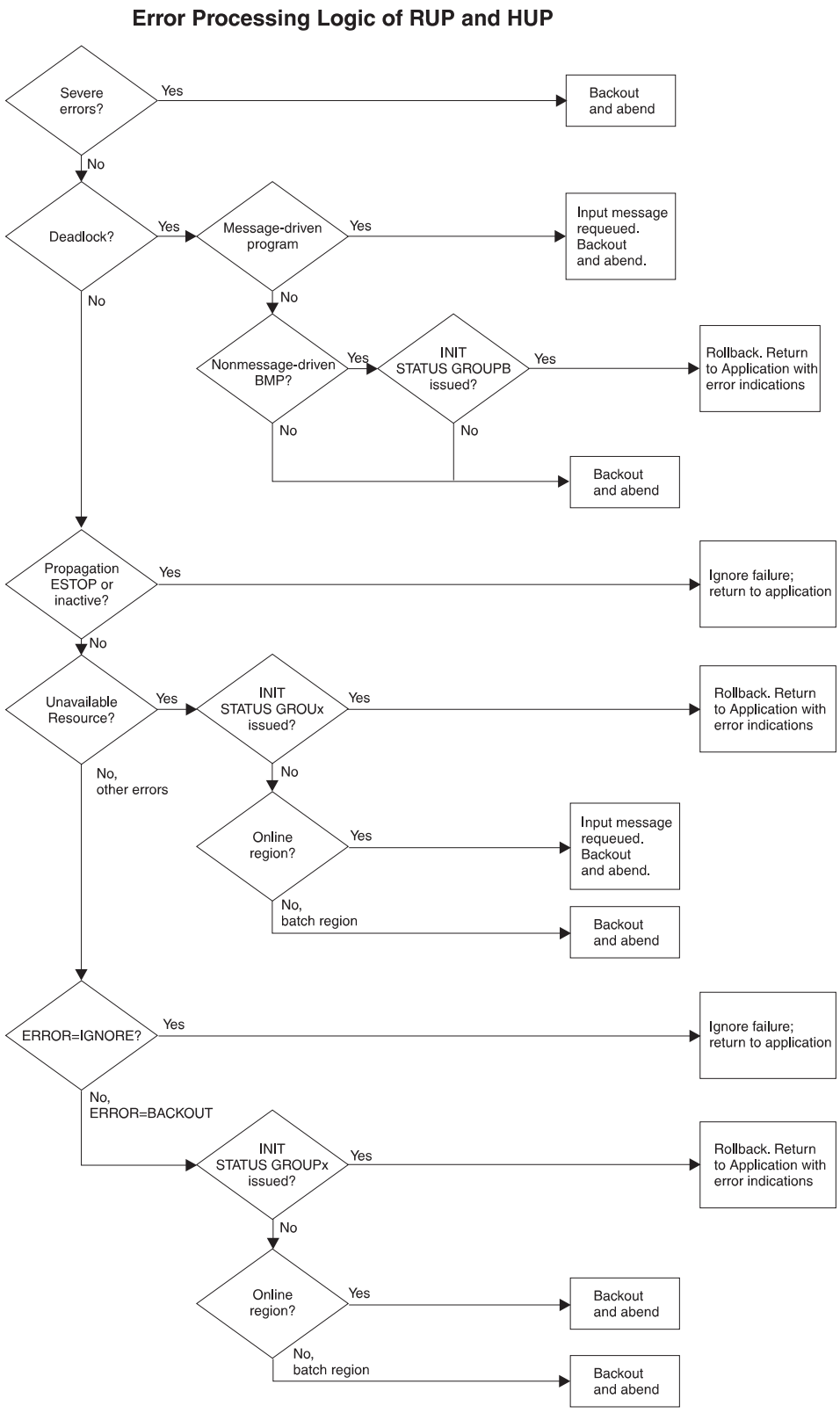


Figure 35. Error Processing Logic of RUP and HUP. Error processing is illustrated from the application program's perspective.

Severe Errors

When a severe error (such as a programming error in an exit routine) occurs, RUP or HUP abends.

DB2 Deadlocks

Because SQL calls issued by RUP and HUP result in DB2 locking activities, a deadlock might occasionally occur.

- If the application program is message-driven, DB2 issues a pseudo-abend to back out uncommitted changes and request requeueing of the input message.
- If the application is a non-message-driven BMP, DB2's IMS attachment facility initiates a rollback and returns control to RUP or HUP.
  - If the application has not issued an IMS INIT STATUS GROUPB call, then RUP and HUP issue an abend.
  - If the application has issued an IMS INIT STATUS GROUPB call, then RUP returns a BC status code and HUP returns a -911 SQL error code ('40000' SQLSTATE).
- In a pure batch environment, DB2's IMS attachment facility rolls back the DB2 changes and abends the region. IMS database changes are not backed out dynamically. To back out changes to the IMS database, you must run the IMS Batch Backout utility (DFSBBO00). For information on running the IMS Batch Backout utility, see *IMS/ESA Utilities Reference: Database Manager*.

IMS Deadlocks

IMS calls issued by HUP result in IMS locking activities that might occasionally cause IMS deadlock.

- If the application program is message-driven, IMS issues a pseudo-abend to back out uncommitted changes and request requeueing of the input message.
- If the application is a non-message driven BMP, IMS backs out all database changes.
  - If the application has not issued an IMS INIT STATUS GROUPB call, then IMS issues an abend.
  - If the application has issued an IMS INIT STATUS GROUPB call, then IMS returns to HUP. HUP returns a -911 SQL error code ('40000' SQLSTATE) to the application.
- In a pure batch environment, IMS and DB2 roll back the IMS and DB2 changes and abend the region.

Propagation Emergency Stopped or Deactivated

When propagation fails and when propagation is emergency stopped for the entire IMS DPROP system, RUP and HUP ignore the propagation failure and return to the application without any error indication and without writing any diagnostic information.

When propagation is deactivated for the failing propagation request, RUP and HUP ignore the propagation failure and continue processing the next propagation request. Or, when there are no more propagation requests, control is returned to the application. No error indication is given to the application when a PR is deactivated.

Unavailable Resources

If a necessary resource is unavailable, RUP or HUP actions vary based on whether the application issued an INIT STATUS GROUPA or GROUPB call.

- If the application issued an INIT STATUS GROUPA or GROUPB call, RUP and HUP issue a ROLB call and return a BB status or -929 SQL error code ('58002' SQLSTATE) to the application.
- If the application program has not issued an INIT STATUS GROUPA or GROUPB call and is running in an online region (MPP or BMP), RUP and HUP issue a ROLS call. The ROLS call:
  - Backs out all uncommitted database changes for the failing UOW
  - Cancels any non-express output messages
  - Puts any input message that the application was processing in the suspend queue for eventual reprocessing
  - Results in a pseudo-abend
- If the application program has not issued an INIT STATUS GROUPA or GROUPB call and is running in a batch region, RUP and HUP issue a ROLB call followed by an abend to take advantage of dynamic backout, if possible. If the ROLB call fails because dynamic backout cannot be called, the application abends.

Other Errors

For other errors, such as non-numeric data in a numeric IMS field, RUP and HUP's actions depend on whether ERROPT for the failed propagation request is specified as IGNORE or BACKOUT.

- For ERROPT=IGNORE, IMS DPROP ignores the propagation failure and returns no error indications to the application. However, RUP and HUP write diagnostic information.
- For ERROPT=BACKOUT:
  - If the application has issued an INIT STATUS GROUPA or GROUPB call, RUP and HUP, regardless of the execution environment, issue a ROLB call and return a BB status or -929 SQL error code ('58002' SQLSTATE) to the application.
  - If the application program has not issued an INIT STATUS GROUPA or GROUPB call and is an online (MPP or BMP) region, RUP and HUP issue an abend.
  - If the application has not issued an INIT STATUS GROUPA or GROUPB call and is in a batch region, RUP and HUP issue a ROLB followed by an abend.

Summary of Error Handling

Table 7 summarizes system error processing for propagation requests when ERROPT=BACKOUT has been specified, showing the error condition the message and non-message driven BMP and the batch regions, for both with and without INIT STATUS call. System error processing for ERROPT=IGNORE is covered in "ERROPT=IGNORE" on page 179.

Table 7. System Error Processing for Propagation Requests Specifying ERROPT=BACKOUT

	WITHOUT INIT STATUS		
ERROR	**MESSAGE DRIVEN MPP/BMP**	**NON-MESSAGE DRIVEN BMP**	**BATCH REGION**[1]

Table 7. System Error Processing for Propagation Requests Specifying ERROPT=BACKOUT (continued)

Deadlock within DB2 or IMS	• DB2/IMS backout occurs • Message is re-queued • Transaction is abended	• DB2/IMS backout occurs • Region is abended	For deadlock in **DB2**: • DB2 backout occurs • Region is abended • Batch backout is required For deadlock in **IMS**: • DB2/IMS backout occurs • Region is abended
Unavailable resource within DB2, IMS, or IMS DPROP	• DB2/IMS backout occurs • Message is re-queued • Transaction is abended	• DB2/IMS backout occurs • Region is abended	• DB2/IMS backout occurs • Region is abended
Other errors	• DB2/IMS backout occurs • Message is discarded • Transaction is abended	• DB2/IMS backout occurs • Region is abended	• DB2/IMS backout occurs • Region is abended
Severe errors	• DB2/IMS backout occurs • Message is discarded • Transaction is abended	• DB2/IMS backout occurs • Region is abended	• DB2/IMS backout occurs • Region is abended
WITH INIT STATUS GROUP*x*			
Deadlock within DB2 or IMS	• DB2/IMS backout occurs • Message is re-queued • Transaction is abended	• DB2/IMS backout occurs • RUP returns BC code • HUP returns -911/40000 code	For deadlock in **DB2**: • DB2 backout occurs • Region is abended • Batch backout is required For deadlock in **IMS**: • DB2/IMS backout occurs • Region is abended
Unavailable resource within DB2, IMS, or IMS DPROP	• DB2/IMS backout occurs • RUP returns BB code • HUP returns -929/58002 code	• DB2/IMS backout occurs • RUP returns BB code • HUP returns -929/58002 code	• DB2/IMS backout occurs • RUP returns BB code • HUP returns -929/58002 code
Other errors	• DB2/IMS backout occurs • RUP returns BB code • HUP returns -929/58002 code	• DB2/IMS backout occurs • RUP returns BB code • HUP returns -929/58002 code	• DB2/IMS backout occurs • RUP returns BB code • HUP returns -929/58002 code
Severe errors	• DB2/IMS backout occurs • Message is discarded • Transaction is abended	• DB2/IMS backout occurs • Region is abended	• DB2/IMS backout occurs • Region is abended

[1.] In batch regions, if dynamic backout (which requires disk data set logging) is not possible the region abends; you must batch backout the IMS data.

Some Causes of Unavailable Resources

When IMS DPROP views a resource as unavailable, errors can occur. This section lists some of the causes of unavailable resources.

- Unavailable DB2 resources:
  - Check pending on a table space (SQL error +162)
  - Incomplete table (SQL error +625)
  - Propagated table has no primary index (SQL error -540)
  - Authority violation (SQL error -551)
  - Authority violation (SQL error -552)
  - Table unavailable (SQL error -653)

  - Cannot execute function in program (SQL error -666)
  - Insufficient virtual storage for buffer pool expansion (SQL error -677)
  - Field procedure could not be loaded (SQL error -682)
  - Program name (DBRM) not found in the plan (SQL error -805)
  - Timestamp mismatch between plan and update module (SQL error -818)
  - Unsuccessful execution caused by unavailable resource (SQL error -904)
  - Resource limit exceeded (SQL error -905)
  - Connection authorization failure (SQL error -922)
  - Connection not established (SQL error -923)
  - DB2 connection internal error (SQL error -924)
  - Not running under DSN (SQL error -927)
- Unavailable IMS resources:
  - Invalid PCB name passed in AIB (RC=X'0104', RS=X'0208')
  - Data management open error (status code AI)
  - BSAM, GSAM, VSAM, or OSAM physical I/O error (status code AO)
  - Call not completed because data was unavailable (status code BA)
  - Call not completed because data was unavailable (status code BB)
  - Database unavailable or has limited availability (status code BK)
  - DEDB inaccessible to request service (status code FH)
  - Segment contains invalid pointer (status code GG)
- IMS DPROP also considers the following events as situations where resources are unavailable for propagation:
  - A propagating application without an appropriate PROP SUSP control statement is running, but propagation is suspended.
  - A propagating application with a PROP SUSP control statement is running, but propagation is not suspended.
  - An application with a PROP OFF control statement is running, but the SCU does not allow use of PROP OFF.

  The PROP OFF and PROP SUSP control statements are discussed in Chapter 13, “Controlling Synchronous Propagation States,” on page 191.

RUP and HUP Error Reporting

After detecting an error, RUP and HUP provides:
- WTO messages to the MVS console
- Error messages to the //EKYPRINT data set
- Error messages to the audit trail
- Error messages to the trace data set, that can be the IMS log, the //EKYLOG data set, or the //EKYTRACE data set
- SNAPs to the trace data set, as appropriate:
  - SNAPs signaled by RUP include interface information between the IMS Data Capture facility and RUP
  - SNAPs signaled by HUP include interface information between the DB2 Data Capture facility and HUP

Limiting the Number of Error Messages Resulting From ERROPT=IGNORE

If you have defined propagation requests with ERROPT=IGNORE, propagation failures can occur and result in many error messages being sent to the MVS console and the audit trail.

You can limit the number of propagation failures that are reported to the MVS console and audit trail. Use the SCU ERRCTL control statement to specify the

maximum number of failures you want reported in keywords of the SCU ERRCTL control statement. The limits only apply to propagation failures when ERROPT=IGNORE is specified and when the failure does not result in an abend or backout. The keywords for SCU ERRCTL are:

MAXPR
Limits the number of failures of individual propagation requests that are reported on the MVS console or the audit trail. MAXPR also limits the number of failures of individual propagation requests that are documented with detailed information in the IMS DPROP trace. You can also set the MAXPR(?) value when creating a propagation request.

MAXSSWTO
Limits the number of failures reported on the MVS console for the entire IMS DPROP system. MAXSSWTO also limits the number of failures documented with detailed information in the IMS DPROP trace for the entire IMS DPROP system.

MAXSSAUD
Limits the number of failures reported on the audit trail for the entire IMS DPROP system.

You cannot use the limits to limit writing of other error messages. The limits describe the maximum number of failures to be documented within a 15-minute interval.

If you know that specific propagation requests often result in propagation failures, you can set the MAXPR value to zero. Then no messages are written to the MVS console and audit trail.

RUP and HUP track how many error messages are written in VLF. If you have not allocated adequate virtual storage for the VLF class used by IMS DPROP in the COFVLF*xx* member of SYS1.PARMLIB, then RUP might not be able to store and retrieve the error counts in VLF, preventing RUP from effectively limiting how many error messages are written. If VLF is unavailable, the number of error messages can be limited only if the MAXPR or MAXSSWTO/MAXSSAUD value is zero.

Even if the specified limits are zero, IMS DPROP writes at least one error message, when a job step encounters propagation failures.

For more information on RUP and HUP's error reporting logic, refer to *Diagnosis*.

Using MVS to Suppress Messages

After running your IMS DPROP system for a while, you might decide you don't want IMS DPROP to issue all the messages to the console. You can use the MVS Message Processing Facility List to suppress writing of detailed messages about propagation failures to the console. The messages are still written to the audit trail and JES log of the propagating job step.

To suppress messages, specify which IMS DPROP message numbers to suppress in the MPFLST*xx* member of SYS1.PARMLIB. Some suggested messages to suppress are:

- EKYR099I
- EKYR600I
- EKYR365I
- EKYR366I

- EKYR367I
- EKYU003E
- EKYX116I
- EKYX117I
- EKYZ360E
- EKYZ360I
- EKYZ380I
- EKYZ381I

Chapter 13. Controlling Synchronous Propagation States

This chapter describes synchronous propagation states and how you can control them using IMS DPROP SCU, RUP, and HUP control statements. This chapter provides a description of:

- Synchronous propagation states and modes
- The status change utility (SCU)
- RUP and HUP control statements

Synchronous Propagation States and Modes

Synchronous propagation states and modes are:

- State of the entire IMS DPROP system
- Status of an individual propagation request
- IMS DPROP PROP OFF mode used for data repair programs

In addition, the following states are relevant when you are doing synchronous propagation:

- IMS read-only status of registered, full-function IMS databases
- DB2 read-only access mode of DB2 table spaces and databases

Synchronous Propagation State of the Entire IMS DPROP System

The status of the entire IMS DPROP system is recorded in the IMS DPROP status file, which is an MVS file, not a DB2 table. Using an MVS file allows RUP and HUP be sensitive to the status of the IMS DPROP system, even if DB2 or the IMS DPROP directory is unavailable.

The IMS DPROP system can be in one of two states:

Emergency Stopped
All synchronous propagation is stopped and all propagation requests are ignored. The status of an IMS DPROP system is set to emergency stopped with the ESTOP control statement of the SCU.

Not Emergency Stopped
Synchronous propagation is done based on the status of individual propagation requests. The state of an IMS DPROP system is set to not emergency stopped with the RESET control statement of the SCU. Unless you specify the DUBIOUS keyword on the RESET statement, RESET deactivates all propagation requests.

Setting the IMS DPROP system to emergency stopped is uncommon. You can set the emergency stopped state when you have severe synchronous propagation problems, for example, a long DB2 outage, and must let updating applications run without synchronous propagation. When you emergency stop an IMS DPROP system, you eventually must resynchronize your IMS and DB2 data.

Synchronous Propagation Status of Individual Propagation Requests

The synchronous propagation status of individual propagation requests is recorded in the IMS DPROP directory and in appropriate VLF objects. An individual propagation request can be only one of the following states:

Inactive
An inactive propagation request does not propagate. You can change the status of a propagation request to inactive by using two SCU control statements: DEACTIVATE and EDEACTIVATE. When first created by the MVG, a propagation request is inactive.

Active
An active propagation request results in propagation as defined in the propagation request. This is the normal operating status of a propagation request. Use the SCU control statement ACTIVATE to change PR status to active.

Suspended
A suspended propagation request does not propagate. Suspending propagation requests can be useful when you are running a few explicitly identified, performance-critical IMS batch or BMP jobs that do many updates and cannot tolerate the increase in elapsed time needed to propagate the updates.

For such performance-critical applications, you might want to:
- Run the jobs without propagating their updates.
- Develop new programs that apply the same updates to the other copy of the data. To reduce elapsed run times, you can run the DB2 and IMS update applications in parallel.

The concept of suspending propagation requests is different from the concept of deactivating propagation requests. If you deactivate a propagation request, the updates of *all* applications are not propagated by the inactive propagation request. If you suspend a propagation request, the updates of only *a few, explicitly designated* jobs are not propagated. You identify the job step for which propagation should not be done by providing a PROP SUSP control statement in the //EKYIN DD statement of the job step.

IMS DPROP does not run PROP SUSP jobs concurrently with jobs without PROP SUSP control statements. At one particular time you can either run jobs with PROP SUSP control statements *or* without, but not both. IMS DPROP protects propagating jobs from encountering failures due to data that is temporarily inconsistent. Failures usually would result in permanent data inconsistencies. (See "PROP SUSP" on page 206 for more information.)

The updates your programs apply to the second data copy should exactly match the updates performed on the first data copy. Updates to the second data copy should be compatible with the mapping logic of IMS DPROP. If you intend to use the CCU to verify data consistency, updates performed on the second copy should be identical to the byte level of fields to the updates resulting from IMS DPROP mapping logic.

PROP OFF Mode for DB Repair Programs

You can turn off synchronous propagation of IMS and SQL update calls for specific jobs using propagation off (PROP OFF) mode. You might turn synchronous propagation off if you use a database repair program that should update only one data copy. An example of database repair is the processing of CCU-generated repair statements.

PROP OFF mode allows you to run repair programs concurrently with normal synchronous propagation activities without impacting normal operations. PROP OFF mode repair programs contain a PROP OFF control statement in the //EKYIN

DD statement of their IMS batch/dependent region JCL. You must authorize PROP OFF mode programs by running SCU with an ALLOWPROPOFF control statement.

Running database repair programs in PROP OFF mode should not be confused with running performance-critical programs in PROP SUSP mode, or deactivating synchronous propagation. Execution of database repair programs with PROP OFF does not affect concurrent updates and synchronous propagation activities. It does not affect the synchronous propagation status of any propagation requests. The synchronous propagation status is not changed from active to inactive or from active to suspended.

Read-Only Status of IMS Databases

Setting IMS databases to read-only mode is important when doing an IMS extract and DB2 load, especially with IMS-to-DB2 and two-way synchronous propagation. You need to make sure that the IMS database is not updated. Read-only mode prevents inconsistencies between the IMS database and the DB2 tables being loaded.

- For full-function IMS databases: If you have implemented DBRC share control and the database is registered in DBRC, you can use SCU to mark the database as read-only and to wait until all IMS systems have released any database update authority. When the SCU returns with a return code of zero, you can assume it has set the database to read-only and that the database is not currently being updated. You can begin the extract and load process in a subsequent job or job step.

  After completing the extract and load and running the associated DB2 utilities (RUNSTATS and COPY), you can call SCU with a READOFF control statement, allowing updates of the database.

  To do the IMS extract and DB2 load:

  1. Run SCU with READON specified to set the database to read-only status in the DBRC RECON data sets.
  2. Do the extract and load, and execute the associated DB2 utilities, such as RUNSTATS and COPY.
  3. Activate any propagation requests that need to be activated.
  4. Run the SCU with READOFF specified to make the database available for updates.

  If DBRC share control is not in effect or the database is not registered, you cannot use the SCU to mark a full-function database as read-only.
- For DEDBs: IMS has no concept of read-only status. Neither IMS nor IMS DPROP gives you an automated way of ensuring the DEDB is not updated while you do the extract and load. You must devise your own method of ensuring the DEDB is not updated during extract and load.

Read-only status is recorded in the IMS RECON data sets, not the IMS DPROP directory or status file.

With two-way synchronous propagation, consider setting the propagated IMS databases in read-only access mode while extracting data with the DLU or for the DLU.

Read-Only Access Mode of DB2 Table Spaces and Databases

Setting DB2 table spaces or databases in read-only access mode is important when doing a DB2 extract and IMS load, especially with DB2-to-IMS and two-way synchronous propagation. Read-only mode prevents inconsistencies between DB2 tables and the IMS database being loaded.

To do the DB2 extract and IMS load:

1. Run the SCU with READON specified to set the affected DB2 table spaces or databases to DB2 read-only access mode. The SCU issues a DB2 -START DATABASE ACCESS(RO) command internally and waits until all active DB2 connections release their update authority. When the SCU returns with a return code of zero, you can assume that it has set the DB2 databases or table spaces to read-only and that they are not being updated.
2. Do the DB2 extract and IMS load process. If you are using the DLU, the DLU checks that the DB2 table spaces or databases being extracted are in read-only mode and that no concurrent DB2 connection can update the tables.

   Complete the extract and load and run any associated IMS utilities, such as Prefix Resolution/Update and Image Copy.
3. Activate any propagation requests that need to be propagated.
4. Run the SCU with READOFF specified to set the DB2 table spaces or databases to DB2 read-write access mode.

   Now you can update DB2 tables and have the updates propagated.

The access mode of DB2 table spaces and databases is recorded in DB2, not in the IMS DPROP directory or status file.

With two-way synchronous propagation, consider setting the propagated DB2 table spaces to read-only mode while extracting the IMS data with DataRefresher or with a user program.

Status Change Utility (SCU)

The SCU allows you control both individual propagation requests and the IMS DPROP system. The SCU also gives you limited control over IMS databases and DB2 table spaces and databases, and provides some general service functions, such as displaying the status of the IMS DPROP system and controlling traces.

The SCU is likely to be your primary tool in controlling and operating your IMS DPROP system. For specific information on running the SCU, see the *IMS DataPropagator Reference*.

If you use IMS DPROP for IMS-to-DB2 synchronous propagation in a production environment, we strongly recommend that you implement DBRC share control and register the propagated IMS databases used for production. By registering your database, you get SCU support for making orderly status changes and setting full-function databases to read-only mode.

The following sections discuss:
- Controlling propagation requests
- Controlling full-function IMS databases
- Controlling DB2 databases and table spaces
- Controlling the IMS DPROP system
- General service functions of the SCU

Controlling Propagation Requests

You can change the status of propagation requests in an orderly and controlled way using the following SCU control statements:

- ACTIVATE changes the propagation request status to active.
- DEACTIVATE changes the propagation request status to inactive.
- SUSPEND changes the propagation request status to suspend. Used with performance critical batch/BMP jobs.
- EDEACTIVATE changes the propagation request status to inactive. Used to deactivate propagation requests in emergency situations.

Changing the Status of Propagation Requests Groups

If you do DB2-to-IMS or IMS-to-DB2 synchronous propagation and have implemented DB2 RIRs between propagated DB2 tables, you should exercise caution when changing the status of propagation requests. As a general rule, give the following propagation requests the same status to avoid synchronous propagation failures:

- All propagation requests propagating to a group of logically related IMS databases, if doing DB2-to-IMS synchronous propagation.
- All propagation requests propagating to a group of tables belonging to the same DB2 referential structure, if doing IMS-to-DB2 synchronous propagation with DB2 RIRs.

  If the status of a propagation request affecting Table A is changed, then all propagation requests for other tables related to Table A should also be changed.

The following scenarios illustrate how you should set the status of propagation requests for IMS-to-DB2 synchronous propagation. For Scenario 1 and 2:

- Segment B is a child of segment A in the same IMS database
- Table B is a child of Table A
- PR1 propagates from segment A to Table A
- PR2 propagates from segment B to Table B

Scenario 1: If PR1 is suspended and PR2 is active, then inserts of segment A are not propagated. Assume that segment B is inserted. It is a child of a segment A, which was also inserted but not propagated. The synchronous propagation of segment B fails because DB2 rejects the insert of segment B since row B has no parent row A.

Scenario 2: If PR1 is active and PR2 is suspended, neither the insert of segment A nor B fails. The insert of segment A propagates successfully to Table A; the insert of segment B does not propagate.

However, depending on the DB2 ON DELETE rule, some deletes of segment A can either:

- Fail, with ON DELETE RESTRICT
- Result in unexpected deletes in Table B, with ON DELETE CASCADE, even though the propagation request propagating to Table B is inactive or suspended.

You might have cases that do not follow these scenarios. If an application updates and propagates ten segments but only one of the segments causes a performance problem, then you might want to suspend synchronous propagation of the one segment if it is the lowest level table in a DB2 RIR.

Making Orderly Status Changes

Except in emergencies, the status of propagation requests should be changed only when the affected data is not being updated. This is called an orderly status change. The status changes done by the following control statements are orderly with one exception: IMS-to-DB2 synchronous propagation when DBRC share control is not in effect or IMS databases are not registered in DBRC.

- ACTIVATE
- DEACTIVATE
- SUSPEND

For status changes between active and suspended and between inactive and active, doing an orderly status change is important for consistency between the IMS and DB2 copy. Doing an orderly change is also important if you want to ensure internal consistency within the data copy that is the target of synchronous propagation.

If you do not do an orderly status change, some updates in a UOW might be propagated while others are not, causing inconsistencies between the IMS and DB2 copies of the data. From the application's point of view, you would have internal inconsistencies within the target data copy.

Emergency deactivation with EDEACTIVATE is not orderly. EDEACTIVATE is executed immediately without waiting for update operations to quiesce, causing inconsistencies.

The SCU supports orderly and controlled status changes for the ACTIVATE, DEACTIVATE, and SUSPEND control statements by:

- **For one-way IMS-to-DB2 synchronous propagation if DBRC share control is in effect and if your IMS databases are registered in DBRC**, making sure status changes performed by the control statements are orderly. The status changes are made when no IMS subsystem has update authority for the affected database.

  To perform an orderly status change, the SCU checks the IMS RECON data sets for each affected IMS database to determine if any IMS subsystem is authorized for update. If so, the operator is informed and the SCU periodically rechecks the authorization and waits until all IMS subsystems have released their update authority.

  – The SCU waits until IMS *online* subsystems release their update authority through an operator-initiated IMS command.

    For registered, full-function IMS databases, when DBRC share control is in effect, you can use the /DBDUMP and /DBRECOVERY control statements to release update authority of an IMS online system. /DBRECOVERY releases the authority faster than /DBDUMP.

    For registered Fast Path DEDBs, when DBRC share control is in effect, you can use the /DBRECOVERY control statement to release the update authority of an IMS online system.

  – The SCU waits until updating *batch* IMS subsystems complete.

    If only IMS batch regions are running, you can use the CHANGE.DB command of DBRC that specify READON or NOAUTH to prevent new updating IMS batch regions from starting.

  After ensuring that no IMS subsystem is authorized for update, the SCU updates the propagation request status in the IMS DPROP directory.

- **For one-way IMS-to-DB2 synchronous propagation if DBRC share control is not in effect or if the affected database is not registered in DBRC**, changing

the status without checking or waiting until IMS subsystems have released their update authority. The SCU completes with a return code of 4. And you must make sure the SCU is called when the affected IMS databases are not being updated in order to avoid inconsistencies between the IMS and DB2 copies and internal inconsistencies within the DB2 copy.

- **For one-way DB2-to-IMS synchronous propagation**, making sure status changes performed by the control statements are orderly. The status changes are made when no DB2 connection is authorized to update the affected DB2 table spaces.

  To perform an orderly status change, the SCU issues a DB2 -DISPLAY DATABASE command internally for each affected table space and checks that no DB2 connection is authorized for update. If a DB2 connection is authorized for update, the operator is informed and SCU periodically rechecks the authorization until all DB2 connections have released their update authority.

  To prevent new DB2 connections from obtaining update authority while SCU is waiting, consider issuing one of the following DB2 commands:

  – -START DATABASE(*database-name*) SPACENAM(*tablespace-name*) ACCESS(RO)

  – -START DATABASE(*database-name*) ACCESS(RO)

  After ensuring that no DB2 connection is authorized for update, SCU updates propagation request status in the IMS DPROP directory.

  You can use propagation requests with unqualified table names to propagate multiple identically-structured DB2 tables having the same unqualified table name. A status change to a propagation request with unqualified table names affects synchronous propagation of all DB2 tables defined for Data Capture that have the same unqualified table name. An orderly status change, therefore, requires that *none* of the tables be updatable while SCU does the requested status change. You might need to make several DB2 table spaces non-updatable, perhaps more DB2 table spaces than you want.

- **For two-way synchronous propagation if DBRC share control is in effect and your IMS databases are registered in DBRC**, making sure status changes done by ACTIVATE, DEACTIVATE, and SUSPEND are orderly.

  The changes are made when no IMS subsystem is authorized to update the affected IMS database, and no DB2 connection is authorized to update the affected table spaces.

- **For two-way synchronous propagation if DBRC share control is not in effect or the affected database is not registered in DBRC**, doing the requested status change without checking or waiting until IMS subsystems have released their update authority. The SCU *does* check that no DB2 connection has update authority for the affected table spaces.

  The SCU completes with a return code of 4. And you must make sure the SCU is called when the affected IMS databases are not being updated.

Activating Propagation Requests

To activate a propagation request for synchronous propagation, use the SCU ACTIVATE control statement. After its creation by MVG/MVGU, a propagation request is inactive. You must run the SCU to activate the propagation request. You also need to activate propagation requests after a previous deactivation or suspension of synchronous propagation.

You can activate a propagation request after you have synchronized IMS and DB2 data and run the appropriate utilities such as the DB2 RUNSTATS and COPY

utilities for IMS-to-DB2 synchronous propagation or the IMS Prefix Resolution/Update and Image Copy utilities for DB2-to-IMS synchronous propagation.

For IMS-to-DB2 synchronous propagation, if DBRC share control is in effect and the affected IMS database is registered in DBRC, the SCU waits until all IMS subsystems release their update authority before changing the status of a propagation request to active.

For DB2-to-IMS synchronous propagation, the SCU waits until all DB2 connections release their table update authority before changing the status of a propagation request to active.

For two-way synchronous propagation, the SCU waits until both IMS and DB2 update authority for the affected IMS databases and DB2 table spaces have been released.

The sequence for activating propagation requests and resetting the read-only mode of the data is:

- For IMS-to-DB2 synchronous propagation. If you previously set the status of an IMS database to read-only and want to restart synchronous propagation for a propagation request, you should *activate the propagation request before making the IMS database available for update operations*. If you activate the propagation request *after* making the database available for updates, some IMS data might be updated but not propagated, resulting in data inconsistencies.
- For DB2-to-IMS synchronous propagation. If you previously set the status of a DB2 table space or database to read-only and want to restart synchronous propagation for a propagation request, you should *activate the propagation request before making the DB2 table space or database available for update operations*. If you activate the propagation request *after* making the data available for updates, some DB2 data might be updated but not propagated, resulting in data inconsistencies.
- For two-way synchronous propagation. If you previously set the status of IMS or DB2 data to read-only and want to restart synchronous propagation for a propagation request, you should *activate the propagation request before making the data available for update operations*. If you activate the propagation request *after* making the data available for updates, some data might be updated but not propagated, resulting in data inconsistencies.

Deactivating and Emergency Deactivating Propagation Requests

To deactivate a propagation request for synchronous propagation, use one of the following SCU control statements:

- DEACTIVATE for orderly deactivation
- EDEACTIVATE for emergency deactivation

For severe synchronous propagation problems, you might decide to deactivate synchronous propagation for selected propagation requests. You also need to deactivate a propagation request before you delete, replace, or rebuild its control information.

How DEACTIVATE Works: Deactivation works differently, depending on the type of propagation.

- For IMS-to-DB2 synchronous propagation, if DBRC share control is in effect and your IMS databases are registered in DBRC, DEACTIVATE does an orderly termination of synchronous propagation activities. SCU waits until all IMS

subsystems with update authority for the IMS databases have released their authorization before changing the propagation request status to inactive. Consistency between IMS and DB2 data is maintained if the data is not updated after synchronous propagation is deactivated.

- For DB2-to-IMS synchronous propagation, the SCU waits until all DB2 connections with update authority for the DB2 table spaces have released their authorization before changing the propagation request status to inactive.
- For two-way synchronous propagation, the SCU waits until both IMS and DB2 update authority for the affected IMS databases and DB2 table spaces have been released.

If propagated data has been updated after a deactivation, you should resynchronize IMS and DB2 data before activating synchronous propagation with an extract and load or CCU-generated repair statements. See Chapter 15, "Verifying Data Consistency (CCU)," on page 219 for information on resynchronizing data. To reactivate synchronous propagation, use the ACTIVATE control statement.

How EDEACTIVATE Works: Emergency deactivation of synchronous propagation does not end synchronous propagation activities in an orderly way. After using this control statement, you might lose consistency between copies of the data. Use emergency deactivation only when consistency of the data copies is less important than the time required to end synchronous propagation. Emergency deactivation might be necessary when an outage has occurred in either IMS DPROP or a database management system, and applications must remain operational and suffer minimum impact from synchronous propagation failures.

SCU deactivates propagation requests without waiting until updating IMS subsystems and DB2 connections complete their updates and release their update authority. Some updates in the same UOW might be propagated while others are not.

It may take a few seconds for RUPs in all propagating IMS regions to be notified of an EDEACTIVATE. The status change in different propagating regions is also not simultaneous.

Suspending Propagation Requests

Suspending selective propagation requests is usually done for a few explicitly identified, performance-critical IMS batch or BMP jobs; these are jobs that usually do many updates and cannot tolerate the increase in elapsed time that might be caused by synchronous propagation.

For such performance-critical applications, you might want to:

- Run the jobs without propagating their updates.
- Develop programs that apply the updates to the other copy of the data. To reduce elapsed run times, you can run the DB2 and IMS update applications in parallel.

You can suspend propagation requests selectively. For example, you can suspend synchronous propagation of updates done by a program to Database A, while updates of the same program to Database B are propagated normally. Refer to "Changing the Status of Propagation Requests Groups" on page 195 for notes and warnings on this subject.

If you have suspended synchronous propagation, you should resynchronize IMS and DB2 data before activating propagation. You usually resynchronize using

programs that apply updates to the other copy of the data; or by an extract and load process; or depending on the propagation request type, by applying CCU-generated repair statements.

You can use the CCU to verify data consistency if you suspend synchronous propagation. The sequence of events for suspending synchronous propagation is:

1. Run the SCU with SUSPEND control statements. Call the SCU when the affected data is not updated.
2. Run your performance critical jobs with PROP SUSP control statements in their //EKYIN DD statement. While the propagation requests are suspended, jobs without PROP SUSP control statements fail if executed.
3. Synchronize the IMS and DB2 data with applications that apply the updates to the other copy of the data. The DB2 and IMS updates can be running in parallel.
4. Reset the propagation request status to active by running the SCU with ACTIVATE control statements. Call the SCU when the affected data is not updated.

Controlling Full-Function IMS Databases

If DBRC share control is in effect, you can use the SCU to control the IMS read-only status of DBRC-registered, full-function databases and prevent inconsistencies during the IMS extract and DB2 load process.

The two following SCU control statements control the read-only status of a registered, full-function database:

- READON sets read-only status
- READOFF resets read-only status to read/write

READON

When processing a READON statement for a registered full-function IMS database, the SCU issues the CHANGE.DB DBD(dbdname) READON command internally. The READON statement marks the database as read-only in the IMS RECON data sets. The SCU then waits until updating batch jobs complete and IMS online systems release their update authority.

Using the SCU to mark a database as read-only has advantages over using other methods to prevent updates of a database. For example, the SCU waits until all IMS subsystems release their update authority before completing its processing. When the SCU returns control with a zero return code, you can assume that the database is not being updated. You can then begin the extract and load process in subsequent job steps.

If you use other methods, such as invoking DBRC with a CHANGE.DB READON command or issuing a /DBDUMP command, you cannot assume that all updates have terminated when the command completes.

READOFF

When calling the SCU with the READOFF statement, which resets the read-only status, make sure all propagation requests that propagate the database are activated before you make the database available for updates.

For specific information on running the SCU and the READON and READOFF control statements, see the *IMS DataPropagator Reference*.

Controlling DB2 Databases and Table Spaces

You can use the SCU to control the read-only status of DB2 databases and table spaces and prevent inconsistencies when extracting from DB2 and loading to IMS during DB2-to-IMS synchronous propagation.

The two following SCU control statements control the read-only status of DB2 databases and table spaces:

- READON sets the DB2 read-only status
- READOFF resets DB2 read-only status to read/write

READON

A READON statement marks one or more DB2 databases or one or more table spaces in the same DB2 database as read-only.

When processing a READON control statement, the SCU internally issues one of the following DB2 commands:

- -START DATABASE(*dbname1*,.....) ACCESS(RO)
- -START DATABASE(*dbname*) SPACENAM(*spacename1*,....) ACCESS(RO)

marking the DB2 databases or table spaces as read-only. The SCU then waits until all DB2 connections release their update authority.

Using the SCU to mark a DB2 database or table space as read-only has advantages over issuing a -START DATABASE ACCESS(RO) command because the SCU waits until all DB2 connections release their update authority. When the SCU returns control with a zero return code, you can assume that DB2 databases or table spaces are not being updated. You can then begin the extract and load process in subsequent job steps.

If you use other methods, such as issuing the -START DATABASE ACCESS(RO) command, you cannot assume that all updates have terminated when the command completes.

READOFF

With one READOFF control statement, you reset the read-only mode of either one DB2 database or one or more table spaces of the same DB2 database.

When calling the SCU with the READOFF statement, make sure all propagation requests that propagate the DB2 database or table spaces are activated before you make DB2 data available for updates.

When processing a READOFF control statement, the SCU sets the specified DB2 table spaces or databases in read-write access mode. The SCU internally issues one of the following DB2 commands:

- -START DATABASE(*dbname1*,.....) ACCESS(RW)
- -START DATABASE(*dbname*) SPACENAM(*spacename1*,....) ACCESS(RW)

For specific information on running the SCU and the READON and READOFF control statements, see the *Reference*.

Controlling the IMS DPROP System

The two following SCU control statements control the state of an entire IMS DPROP system:

- ESTOP sets the emergency stopped state
- RESET sets the not emergency stopped state

ESTOP

The emergency stop (ESTOP) statement allows existing IMS applications to proceed even when DB2 resources, including the IMS DPROP directory or the DB2 system itself, are not available.

ESTOP does not wait for updating subsystems to release their update authority for the affected IMS databases. It might take a few seconds for RUPs in all propagating IMS regions to be notified that the IMS DPROP system has been emergency stopped. The status change in different propagating regions is not simultaneous.

If you emergency stop IMS DPROP, you might need to resynchronize IMS and DB2 data before starting synchronous propagation. Avoid emergency stopping IMS DPROP; it can result in inconsistencies between all propagated databases and tables.

As an alternative to ESTOP, if the DB2 system parameter DPROP SUPPORT has been set to 3, you can also stop all DB2-to-IMS synchronous propagation activities using the DB2 command *-STOP TRACE(MONITOR) CLASS(6)*. This alternative is not recommended, because IMS DPROP neither notices nor records that synchronous propagation has been stopped and that updates have not been propagated.

RESET

When you decide to resume synchronous propagation, you must reset the emergency stopped status of the IMS DPROP system with a RESET statement.

- Usually you issue RESET without the DUBIOUS keyword. The SCU first marks all propagation requests in the IMS DPROP directory as inactive and then marks the IMS DPROP system in the status file as not emergency stopped.

  If the IMS and DB2 data have become inconsistent, you need to resynchronize your data copies. When resynchronization is complete, you can activate the propagation requests.
- In some cases, you might issue a RESET control statement with the DUBIOUS keyword. With DUBIOUS, the SCU indicates in the status file that the IMS DPROP system is not emergency stopped; the SCU does not change the status of the propagation requests. Synchronous propagation resumes immediately without a prior resynchronization of data.

  Use DUBIOUS when you can take the risk of inconsistent data, resume synchronous propagation, and defer the resynchronization of IMS and DB2 data until a later time. Use this option with caution; synchronous propagation can fail because of inconsistent data.

  It may take a few seconds until RUPs in all propagating IMS regions are notified of the RESET DUBIOUS statement. The status change in different propagating regions is not simultaneous.

See Chapter 15, “Verifying Data Consistency (CCU),” on page 219 for information on resynchronizing data. For specific information on ESTOP and RESET, see the *Reference*.

General Service Functions of the SCU

The SCU has several other functions you might find useful for administering the IMS DPROP system. This section covers:

- Turning propagation off using ALLOWPROPOFF and DENYPROPOFF
- Displaying system information

- Changing error options using ERROPT
- Changing error control information using ERRCTL
- Initializing the IMS DPROP system, status file, and VLF objects
- Turning tracing on and off

Turning Synchronous Propagation Off Using ALLOWPROPOFF and DENYPROPOFF

The SCU can protect against inadvertent execution of database repair jobs that turn off synchronous propagation of IMS and DB2 updates. This protection requires that an authorized user (one with the DB2 privilege to execute the SCU plan) call the SCU using the ALLOWPROPOFF control statement. ALLOWPROPOFF allows updating IMS regions to turn off synchronous propagation of their IMS database and DB2 table updates by including the PROP OFF control statement in the //EKYIN DD of their JCL.

Running database repair programs in PROP OFF mode should not be confused with running performance-critical programs in PROP SUSP mode, or deactivating synchronous propagation. Execution of database repair programs with PROP OFF does not affect concurrent updates and synchronous propagation activities. It does not affect the synchronous propagation status of any propagation requests. Synchronous propagation status is not changed from active to inactive or from active to suspended.

After the job steps complete, call the SCU again with DENYPROPOFF to deny execution of database repair programs. DENYPROPOFF prevents propagating IMS regions from turning off synchronous propagation.

The sequence of events for running database repair programs with PROP OFF control statements is:

1. Run the SCU with ALLOWPROPOFF.
2. Run your database repair program with a PROP OFF control statement in its //EKYIN DD.
3. To deny subsequent execution of PROP OFF mode programs, run the SCU with DENYPROPOFF.

You use the SCU to *allow* synchronous propagation to be turned off, but you must use the RUP control statement PROP OFF to actually turn synchronous propagation *off*.

Displaying System Information using DISPLAY, LIST.DB, -DISPLAY DATABASE

Use the DISPLAY control statement to see:

- Whether the system has been emergency stopped
- The approximate amount of virtual storage required by VLF system objects

Directory information is stored in relational tables and can be accessed and displayed with QMF queries.

The LIST.DB command of DBRC displays the status of IMS databases. This command is documented in *IMS/ESA Utilities Reference: Database Manager*.

The -DISPLAY DATABASE command of DB2 displays the status of DB2 databases and table spaces. This command is documented in *DB2 Command Reference*.

Changing Error Options Using ERROPT

When you create a propagation request, you can specify an error option of IGNORE or BACKOUT. This specification determines how RUP and HUP handle data synchronous propagation failures and other errors when encountered. More information about these error options and their relationship to the error handling process of RUP and HUP is presented in "Error Handling Options" on page 177.

To change the PR error option, use the SCU ERROPT control statement.

Changing Error Control Information Using ERRCTL

If you have defined propagation requests with ERROPT=IGNORE, propagation failures can occur and result in many error messages being sent to the MVS console and the audit trail.

You can limit the number of propagation failures that are reported to the MVS console and audit trail. Use the SCU ERRCTL control statement to specify the maximum number of failures you want reported in keywords of the SCU ERRCTL control statement. The limits only apply to propagation failures when ERROPT=IGNORE is specified and when the failure does not result in an abend or backout. The keywords for SCU ERRCTL are:

MAXPR
Limits the number of individual propagation request failures reported on the MVS console and audit trail.

MAXSSWTO
Limits the number of failures reported on the MVS console for the entire IMS DPROP system.

MAXSSAUD
Limits the number of failures reported on the audit trail for the entire IMS DPROP system.

You cannot use the limits to limit writing of other error messages. The limits describe the maximum number of failures to be documented within a 15-minute interval.

If you do not want your console to receive many messages, you can set low values for these limits. If you want all error messages written to the console or audit trail, you can specify UNLIMITED for the error message limits.

You can use the NEWCYCLE keyword of the ERRCTL control statement to force RUP and HUP to resume writing error messages when you are in the process of fixing synchronous propagation failures. You can then see if failures are still occurring.

Initializing the IMS DPROP System, Status File, and VLF Objects (INIT)

Using the SCU you can initialize:
- IMS DPROP system, both directory tables and status file
- IMS DPROP status file only
- VLF objects

IMS DPROP System: When you are customizing your IMS DPROP system, you must initialize both the IMS DPROP directory and status file.

IMS DPROP Status File: If the status file is lost or destroyed, you can rebuild it with the INIT STATF control statement. If the IMS DPROP system is emergency stopped when you rebuilt the status file, also include an ESTOPPED keyword on the control statement.

VLF Objects: Use the INIT VLF control statement to build all VLF objects that contain IMS DPROP control information. You usually would use INIT VLF after an IPL or deactivation of VLF. It is better to rebuild VLF objects with INIT than when RUP, HUP, and other IMS DPROP components need to access them. You avoid potential enqueuing problems on the IMS DPROP directory tables. RUP and HUP control statements are discussed in "RUP and HUP Control Statements" on page 205.

You also use the INIT VLF control statement after operational or software errors have created the wrong objects in VLF.

Turning Tracing On and Off (TRACEON, TRACEOFF)

You can use the SCU to turn the tracing of synchronous propagation activities on and off. You might do this when you are debugging and implementing propagation requests. Tracing is started using the SCU TRACEON control statement; TRACEOFF stops tracing.

Starting the trace with SCU allows you to trace synchronous propagation activities on a system-wide basis. Starting the trace with RUP and HUP control statements in //EKYIN allows you to trace the activities of individual batch or dependent regions.

RUP and HUP Control Statements

RUP and HUP control statements are input with the //EKYIN DD statement in your IMS batch or dependent region JCL. Some of these control statements operate by themselves, while others work with the SCU. The control statements control synchronous propagation and tracing information.

RUP and HUP control statements are:
- PROP LOAD, used only for IMS-to-DB2 synchronous propagation
- PROP OFF
- PROP SUSP
- TRACE
- TRDEST
- RESIDENT

You can use TRACE and TRDEST for all types of IMS DPROP job steps:
- Batch and dependent IMS regions doing synchronous propagation
- Job steps used to call RUP for asynchronous propagation
- IMS DPROP utilities

You use the other control statements only for IMS regions doing synchronous propagation.

For syntax diagrams and detailed information on these control statements, refer to the *Reference*. The following subsections discuss:
- Controlling synchronous propagation using PROP control statements
- Controlling traces
- Controlling the number of resident SQL update modules and PRCBs

Controlling Synchronous Propagation Using PROP Control Statements

You can control synchronous propagation using the PROP LOAD, PROP OFF, and PROP SUSP control statements. PROP control statements are mutually exclusive; only one control statement can be coded in the //EKYIN DD statement.

PROP LOAD

Use PROP LOAD for only *IMS-to-DB2* synchronous propagation.

By default, RUP does not propagate IMS segments inserted through PCBs specifying a PROCOPT of L or LS. If you want to propagate these insert calls, you must override IMS DPROP's default using a PROP LOAD control statement; PROP LOAD is provided in the EKYIN data set of the IMS job steps loading the IMS database.

When doing asynchronous propagation, RUP always propagates insert calls, including those specified with a PROCOPT of L or LS. If there are IMS insert calls with PROCOPT=L that you do not want to propagate when doing user asynchronous propagation, then your sender or receiver program should suppress their synchronous propagation.

PROP OFF

Use PROP OFF to turn off synchronous propagation of IMS and SQL update calls issued by database repair programs.

To use PROP OFF, you must first run the SCU with an ALLOWPROPOFF control statement. After completing your database repair program, run the SCU again with DENYPROPOFF specified to prevent execution of other jobs with PROP OFF control statements.

PROP SUSP

Use PROP SUSP to identify performance critical jobs that cannot tolerate the performance impact of synchronous propagation.

Before running jobs with PROP SUSP, you must first run the SCU with SUSPEND control statements. SCU SUSPEND sets the propagation request status in the IMS DPROP directory to suspended.

While processing an IMS update for a particular segment type, RUP verifies whether the status of all propagation requests for IMS-to-DB2 synchronous propagation of the segment type are compatible with the provided //EKYIN control statements. Similarly, while processing a DB2 update for a particular table, HUP verifies whether the status of all propagation requests for DB2-to-IMS synchronous propagation of the table are compatible with the provided //EKYIN control statements.

If the propagation request status is suspended, RUP and HUP prevent updates made without an appropriate PROP SUSP or PROP OFF control statement. If the propagation request status is active, RUP and HUP prevent updates made with a PROP SUSP control statement in the //EKYIN DD. Table 8 on page 207 provides more detail.

If you suspend synchronous propagation, you must resynchronize the IMS data and its DB2 copy.

Although PROP SUSP is intended for use with batch and BMP applications, you can use it in IMS message regions. You can isolate performance critical MPPs in their own message regions and use PROP SUSP in the //EKYIN DD statements of these message regions. Make sure that the MPPs are executed in the correct message regions.

Relationship of PR Status and PROP SUSP/OFF Control Statements

Table 8 describes the results of combining:

- Propagation request status in the IMS DPROP directory
- PROP SUSP/OFF control statements

A conflict between the propagation request status and the control statements is treated by RUP and HUP as an error. RUP and HUP apply their error handling logic for unavailable resources and back out the updates performed in the UOW.

Table 8. Update Actions Based on Propagation Request Status and PROP Control Statements

PR Status in IMS DPROP Directory	RUP/HUP Control Statement in //EKYIN	Updates Allowed	Updates Propagated
Active	None	Yes	Yes
	PROP SUSP	No/Backout	N/A
	PROP OFF	Yes*	No
Inactive	Ignored	Yes	No
Suspended	None	No/Backout	N/A
	PROP SUSP	Yes	No
	PROP OFF	Yes*	No

* Assuming the SCU ran with an ALLOWPROPOFF control statement and that IMS DPROP is not emergency stopped

Controlling Traces

Use TRACE and TRDEST control statements to start tracing synchronous propagation activities for a particular job. For synchronous propagation, you can also start the trace with the SCU, but the trace will start on a system-wide basis, whereas the RUP and HUP control statements apply to only a single region.

TRACE

Use TRACE to start tracing propagation activities for a job step. You can also use TRACE with the SCU tracing function. The information provided by both trace methods is the same.

Depending on the level of trace selected, you can receive a large volume of trace data. You should be aware of the impact tracing can have on the IMS log, or allocate an adequate amount of space for the //EKYTRACE or //EKYLOG data set. The options available on the TRACE control statement are described in the *Reference*.

You can supply multiple TRACE control statements in a single //EKYIN DD statement. For instance, you might want to trace synchronous propagation activities for various databases at different trace levels.

TRDEST

Use TRDEST to direct the output of a trace to a particular destination.

In IMS batch and dependent regions doing synchronous propagation, use TRDEST to direct trace output to the:

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

If you do not specify TRDEST to RUP and HUP, trace data is written to the IMS log.

Trace output to the IMS log or to //EKYLOG is unformatted and, therefore, requires fewer I/O operations, less external storage, and less CPU overhead for the traced job step. Writing trace output to the IMS log or //EKYLOG also allows you to print and format trace records selectively.

To print IMS DPROP trace records written to the IMS log or to //EKYLOG, IMS DPROP has a formatting and printing routine (EKYZ620X). The routine runs as an exit to the IMS File Select and Formatting Print utility (DFSERA10). You use the typical DFSERA10 JCL and control statements to select IMS DPROP trace records from the IMS log or from //EKYLOG. On these control statements, you code an EXITR keyword that specifies EKYZ620X. For additional information about this routine, refer to *Diagnosis*. More information on running DFSERA10 is in *IMS/ESA Utilities Reference: System and IMS/ESA Utilities Reference: Database Manager*.

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

Controlling the Number of Resident SQL Update Modules and PRCBs

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

- SQL update modules that remain resident in the virtual storage of each IMS region during IMS-to-DB2 synchronous propagation
- RUP control blocks (PRCBs) that remain resident in the virtual storage of each IMS region during IMS-to-DB2 synchronous propagation
- HUP PRCBs that remain resident in the virtual storage of each IMS region during DB2-to-IMS synchronous propagation

Resident SQL Update Modules

In MPP regions, RUP might frequently load SQL update modules that have been generated for each propagation request belonging to a generalized mapping case. Performance measurements show that the CPU time required to issue MVS LOAD macros to load these modules can be substantial in MPP regions.

RUP contains logic that maintains in the virtual storage of each propagating region:

- All SQL update modules used during the current program execution.
- The SQL update modules associated with RUP PRCBs that are not used by the current MPP execution, but which are nevertheless resident in the virtual storage of the propagating region.

- The most recently used SQL update modules that have been used during previous MPP executions. The default for number of recent modules is 40. Depending on the number of propagation requests and SQL update modules executed in an MPP region, you might want to specify a higher or lower number than the default value of 40.

  You can override the default value of 40 using a RESIDENT control statement in the //EKYIN data set of the propagating region.

Resident PRCBs

In MPP regions, RUP and HUP might frequently read PRCBs describing the mapping and synchronous propagation. Performance measurements show that you can achieve that moderate improvements in CPU time by avoiding the path length required to read the PRCBs from VLF in MPP regions.

RUP and HUP therefore contain logic that maintains in the virtual storage of each propagating region:

- All PRCBs used during the current program execution.
- The most recently used PRCBs from prior MPP executions. The default number of prior executions is 20. Depending on the number of segment types and tables that are propagated in an MPP region, you might want to specify a higher or lower number than the default value of 20.

  You can override the default value of 20 using a RESIDENT control statement in the //EKYIN data set of the MPP region.

Chapter 14. Database Maintenance for Synchronous Propagation

This chapter assists you when you are doing database maintenance while running synchronous propagation. (Refer to the *IMS DataPropagator Customization Guide* for information on database maintenance while doing asynchronous propagation.) Proper database maintenance helps keep the IMS and DB2 data copies in synchronization. Topics described in the chapter include:

- IMS and DB2 checkpoint and restart
- Database backout
- System data set and database backup and recovery
- Timestamp recovery
- Data resynchronization
- Database repair
- Database reorganization and load considerations
- CCU verification
- IMS DPROP directory recovery

Some general considerations on database maintenance are:

- IMS-to-DB2 propagation. When recovering or reorganizing an IMS database, do not update the database. Make the database unavailable for updates to prevent propagation. The IMS utilities and other facilities[12] that do not use standard DL/I calls to access IMS data will not propagate data.
- DB2-to-IMS propagation. When recovering or reorganizing in DB2, do not update the tables. Make all affected DB2 table spaces unavailable for updates to prevent propagation. Similar to IMS, DB2 utilities such as LOAD, RECOVER, REORG, and REPAIR do not propagate data to IMS.
- Two-way propagation. Make IMS databases and DB2 table spaces unavailable for update.
- Use of DBRC. We recommend that all IMS subsystems used for production work run under DBRC share control and that propagated IMS databases be registered in RECON. All IMS subsystems propagating to the same DB2 should use the same RECON.
- Synchronization points and data integrity. When propagating data synchronously, RUP and HUP run in the same address space as the propagating IMS application. From an IMS perspective, a propagating application is considered mixed-mode. Because of the two-phase commit process controlled by the IMS synchronization point manager, both IMS and DB2 can maintain data integrity across system failures and subsequent recovery and restarts.

Checkpoint and Restart in the IMS and DB2 Environment

IMS considers a propagating application a mixed-mode transaction. Because of the two-phase commit process controlled by IMS, both IMS and DB2 can maintain data integrity across system failures and subsequent recovery and restarts if both IMS and DB2 have been reactivated through a warm start.

12. Examples of such facilities include the IMS Utility Control Facility (UCF) as well as the VSAM Zapper and Fast Reorganization Reload programs, which are part of the Database Tools (DBT) program product.

Restart of IMS Online and DB2

If IMS requires a cold start, you should always try to perform a normal restart of DB2, rather than a conditional or cold start of DB2.

After a cold start of IMS, there is no synchronization between the IMS and DB2 database buffer manager. You risk mismatches between propagated IMS and DB2 data, and you must at least check consistency using the CCU. A cold start of DB2 might reduce, but not eliminate, mismatches between IMS and DB2 in a data propagation environment; however, the cold start usually causes inconsistencies within DB2 objects. Consider cold starting a DB2 or IMS system only if your attempt to warm start has failed.

Checkpoint and Restart of an IMS Batch Program

As described in "IMS Application Checkpoint and Restart" on page 85, propagating programs must issue IMS checkpoint calls. Propagating batch programs should also issue a final checkpoint call after making the last IMS update call.

Propagating batch programs should allow restart from the last checkpoint issued.

Also, DB2 requires that the job name of a restart job be the same as the job name of the failing propagating batch job.

Database Backout for IMS Batch Programs

One way to restore database integrity after an application error is to back out the application's updates to the database. This section examines some of the implications of database backouts when doing synchronous propagation:

- IMS dynamic backout for batch regions
- Backout of committed data

IMS Dynamic Backout for Batch Regions

If a propagating IMS batch program abends and you cannot dynamically back out the IMS updates, then only the DB2 changes are automatically backed out. To allow dynamic backout of IMS changes to occur, you should log to a direct-access device and specify BKO=Y in the JCL of the propagating IMS batch job. Otherwise, you must run the IMS Batch Backout utility before restarting the abended job.

More information on how to specify dynamic backout when using the batch IMS JCL procedure (DLIBATCH or DBBBATCH) is in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Propagation failures and IMS pseudo-abends are types of errors that are dynamically backed out. Therefore, we strongly recommend that you use dynamic backout in propagating IMS batch jobs. Dynamic backout is easier to perform than the Batch Backout utility, and also expedites restart of abending applications.

Application program abends such as program checks (for example S0C1) are not dynamically backed out. For more information on how to run the IMS Batch Backout utility (DFSBBO00), refer to *IMS/ESA Utilities Reference: Database Manager*.

Backout of Committed Data

You cannot back out committed data in DB2 the way you can in IMS with the IMS Batch Backout utility (DFSBBO00).

You cannot backout DB2 changes made before the last commit point. Therefore, you should not use the Batch Backout utility to back out IMS changes made before the last commit point. IMS prevents backout when the batch job has been run with the IMS resource lock manager (IRLM).

If you must back out the effect of a batch job on IMS data before the last commit point, you must resynchronize DB2 data either by:
- Running a point-in-time recovery
- Re-extracting the data from IMS and reloading to DB2
- Running the CCU and generating and applying the repair file

Backup and Recovery

When implementing data propagation, you need to update your backup and recovery procedures. Automate your procedures, if possible.

System Data Sets

Backup of vital data in IMS (such as the RECON data sets) and DB2 (such as the BSDS and DB2 catalog) should be run concurrently to establish a common point of consistency and simplify operations for IMS and DB2 recoveries and restarts. If one of the data sets is lost, both IMS and DB2 might need to be restored from the backups. If backup of both system's data is done at the same time, it is easier to keep the systems synchronized.

Databases

You can run save and backup procedures independently for IMS and DB2 databases, as long as you can run normal recovery procedures. However, you should consider running concurrent backups periodically to provide a point in time when both systems have a common recovery point or point of consistency. To run concurrent backups in an IMS-to-DB2 propagation environment, make the propagating IMS database and then its DB2 counterpart unavailable for update. Then make an image copy of both sides. For DB2, you can use either the incremental image copy option or the full image copy option. After making the image copies, you can make the DB2 side available for updating, followed by the IMS side.

The same process applies for DB2-to-IMS propagation. Make the propagating DB2 table space and then its IMS counterpart unavailable for update. Then back up both sides. Finally make the IMS and then the DB2 side available for updates.

You can also use concurrent quiesce points to establish a common recovery point. The quiesce points are marked on the IMS side by issuing a /DBR command without the NOFEOV keyword. Quiesce points are marked on the DB2 side by running the Quiesce utility, which sets a quiesce point on the DB2 table space. You can find an IMS quiesce point in RECON using a time stamp; you can find a DB2 quiesce point in the DB2 catalog using the relative byte address (RBA).

If you have to recover the IMS side by restoring it from the database image copy, then you should also restore the DB2 side from the corresponding image copy of the DB2 table space. And vice versa. You can ensure both sides are consistent without having to run the CCU or an extract and load of the IMS or DB2 side. If no common recovery point exists or if this process does not work, then you have to do an extract and load or run the CCU.

For DB2, it is good practice to make an image copy simultaneously of all table spaces involved in an RIR. For IMS, you should make an image copy simultaneously of all databases involved in a logical relationship.

Timestamp Recovery

Point-in-time (timestamp) recovery for database-related objects being propagated is not recommended except for special cases, such as an abend of a long-running IMS batch program.

IMS supports timestamp recovery. For DB2, you can recover to a point when an incremental or full image copy was taken. However, there is no automated way to recover both IMS and DB2 to the same point in time.

IMS and DB2 recoveries must be synchronized. On the IMS side, you can start timestamp recoveries from timestamps that you select and IMS approves. On the DB2 side, you must select relative byte address (RBA) numbers or selected backup copies. You have to specify, in IMS and DB2 terms, the point in time to which affected databases should be restored.

After performing an IMS timestamp recovery, you might want to do a new extract and load of the DB2 tables rather than recover the propagated DB2 tables. However, extract and load might be a lengthy process and unacceptable for large tables. When propagating DB2 data to IMS, you might also use the DLU to re-create the IMS copy from DB2 data that has been timestamp recovered to a selected quiesce point.

Data Resynchronization

If IMS-to-DB2 propagation has been deactivated or suspended, you must resynchronize the propagated data in DB2 with the IMS data. You can either:

- Re-extract the data from IMS and reload it into DB2. If possible, program resynchronization jobs to run automatically. Your automated jobs will usually consist of an IMS unload or DataRefresher extract, DB2 load, and control steps.
- Run the CCU and use the file of SQL corrections it generates as input to DSNTEP2 or DSNTIAD.

  In some cases, if there are few changes necessary, it might be more efficient to run the CCU instead of an extract and load, and apply the repair changes generated by the CCU to achieve consistency. Limit use of the CCU to propagation requests using the generalized mapping cases.

Regardless of which alternative you use, make the database unavailable for update on the IMS side so consistency can be established. Use the SCU and a READON control statement to prevent database update. After resynchronization is complete, restart propagation. Then make the database available for update using the SCU READOFF control statement.

Similar considerations apply for DB2-to-IMS propagation when propagation has been deactivated or suspended. In order to resynchronize the propagated data, you can either:

- Run the IMS HD Reorganization Unload utility (DFSURGU0) if not all IMS data is subject to propagation. The DLU uses the sequential data set that is created. Then use the DLU to load all IMS database data from the DB2 copy and, if necessary, from the sequential input data set. Finally, save the IMS database using the IMS Image Copy utility.

- Run the CCU, and apply the file containing the DL/I repair statements generated by the CCU to the IMS database; use the DL/I test program (DFSDDLT0).

Before resynchronizing data, make the DB2 side unavailable for update by using the SCU and a READON DB2DB control statement. After resynchronization, you can reactivate propagation. Then, you can make the DB2 side available for updates by using the SCU and a READOFF DB2DB control statement.

Database Repair

You can repair databases using IMS and DB2 repair functions or user-written programs. This section discusses:
- IMS and DB2 repair functions
- User-written repair programs
- Preventing inadvertent execution of repair programs

IMS and DB2 Repair Functions

You can repair bad pointers and user data in IMS databases using the ZAP function of the Utility Control Facility (UCF) of IMS/ESA. Similar functions are provided for VSAM databases by the VSAM Zapper program of the Database Tools (DBT) program product. For repairing DB2 table spaces, you can use the DB2 Repair utility.

When repairing an IMS or DB2 database, it is your responsibility to decide whether the database and its counterpart should be made unavailable for read-only or for update by other programs.

In most cases, you make an IMS database unavailable for access by issuing a /STO DATABASE or /DBR DATABASE operator command. After the repair is done, you can restart the database using a /STA DATABASE command.

For DB2, you can make the table space that needs to be repaired inaccessible to any SQL statements by issuing a -START DATABASE ACCESS(UT) command. Then, after the repair is done, you can restart using data propagation. Then you can make the DB2 side accessible by issuing a -START DATABASE ACCESS(RW) command.

User-Written Repair Programs

The SCU allows you to run repair programs that you write on the IMS or DB2. If an IMS database or a DB2 table space has to be repaired using programs you write, the SCU allows you to run the repair program on the IMS or DB2 side that does not propagate the repairing DL/I calls or SQL statements to the other side.

To set propagation mode off:
1. Call the SCU specifying ALLOWPROPOFF.
2. Run the repair job specifying PROP OFF in the //EKYIN DD statement.
3. Call the SCU specifying DENYPROPOFF.

The propagation off mode allows you to run repair programs concurrently with normal propagation activities without impacting normal operations. You do this by including a PROP OFF control statement in the //EKYIN DD statement of the job stream for the application whose propagation you are stopping.

Use the ALLOWPROPOFF control statement to enable the SCU to use PROP OFF. You then can run updating job steps when PROP OFF is specified.

Running database repair programs with PROP OFF control statements is different from running performance-critical programs in propagation suspended (PROP SUSP) mode. It is also different from deactivating propagation. Running database repair programs with PROP OFF control statements does not affect other concurrent updates and propagation activities. It does not affect the propagation status of any propagation requests. Propagation status is not changed from active to inactive or from active to suspended.

After running the database repair job step, you usually call the SCU again, specifying DENYPROPOFF control statements to prevent additional database repair programs from running.

For more information on coding ALLOWPROPOFF and DENYPROPOFF and running the SCU, refer to the *Reference*.

Preventing Inadvertent Execution of Repair Programs

You can use the SCU with PROP OFF control statements to prevent inadvertent execution of database repair programs or other programs. An authorized user (one with the DB2 privilege to execute the DB2 plan of the SCU) calls the SCU specifying the ALLOWPROPOFF control statement.

Database Reorganization and Load

You can reorganize IMS databases and DB2 table spaces independently of each other. You can also perform normal recovery independently in IMS or DB2 when I/O errors are encountered. However, when IMS-to-DB2 synchronous propagation is active and the DB2 side is reorganized, you should deactivate propagation or stop the propagating IMS transactions/databases. If you do not deactivate or stop propagation, propagating IMS applications abend when required DB2 resources are unavailable. Similarly, during DB2-to-IMS propagation, you should deactivate propagation or stop the propagating IMS transactions when the IMS side is reorganized.

During a reorganization, *no* additional changes or deletions should be made during the unload phase of the process. IMS segments or DB2 rows dropped at unload and reload time are transparent to IMS DPROP. If IMS segments are dropped during a database reorganization, you must reflect the dropped data in the DB2 tables by resynchronizing DB2 table spaces with the IMS database. Also, if DB2 rows are dropped during a table space reorganization, then you must reflect the dropped data in the corresponding IMS databases by resynchronizing IMS databases with DB2 table spaces.

Instead of reorganizing IMS or DB2, you can re-extract data and reload it. This may be faster than reorganizing IMS or DB2. You can also reorganize both the IMS and DB2 sides in parallel rather than sequentially to reduce the time it takes to reorganize both systems.

Make sure procedures for space control of propagated DB2 table spaces and IMS databases are in place before implementing propagation on the production system.

Initial Load of IMS Databases

For IMS, the initial load of a database must be done using a user-written program, which requires a PSB. When using a PROCOPT of L or LS, propagation is not done unless explicitly requested. If you want propagation, you must explicitly request it using a PROP LOAD control statement in the JCL of the loading job step. If your program is used to load an IMS database, then propagation to DB2 can take place concurrently. DataRefresher has an alternative for the initial extract and load of the data from IMS to DB2. Performance could be poor for large tables since rows are not inserted into the DB2 tables through the DB2 load interface.

Load of DB2 Tables

For DB2, you usually load a table using the DB2 load utility. However, as with other DB2 utilities, the DB2 load utility does not propagate data to IMS.

When doing DB2-to-IMS propagation, you usually must resynchronize the IMS copy with the new DB2 copy after the DB2 load, using the DLU.

CCU Verification

After performing maintenance on databases involved in propagation, run the CCU to verify consistency of the propagated data. Also run the CCU after you make image copies to verify consistency of the image copies and so that you have a known point of consistency in recovery activities.

For information on the CCU, see Chapter 15, "Verifying Data Consistency (CCU)," on page 219 and the *Reference*.

IMS DPROP Directory Recovery

When recovering IMS DPROP directory tables, you usually recover the tables to their *current state*. You probably won't need to recover directory tables to a previous state or point in time. If you do recover the directory tables to a previous point in time, you should verify consistency of:

- Content of all IMS DPROP directory tables
- IMS DPROP status file
- VLF copy of IMS DPROP control information
- Libraries containing the SQL update modules
- Libraries containing the DBRMs of the SQL update modules
- DB2 plans of propagating application programs
- Definitions of the IMS database in the IMS DBDLIB and ACBLIB, and in the DataRefresher FDTLIB and EXTLIB
- IMS databases
- Definitions of the DB2 tables in the DB2 catalog
- DB2 tables

When recovering a directory table space to a previous point in time:

- If you have directory tables in more than one table space, recover each directory table space to the same point in time. This is required only if the directory table spaces have changed since that point in time.
- The IMS DPROP status file might become inconsistent with the IMS DPROP directory if the SCU ran between the recovery point and the present time and one of the following control statements is present:
  - INIT DPROP
  - ESTOP

– RESET

If the IMS DPROP status file and IMS DPROP directory are inconsistent, you must call the SCU using an INIT STATF control statement.

- After completing the recovery, call the SCU using an INIT VLF control statement to resynchronize the directory with the VLF copy of IMS DPROP control information.
- If you have created or replaced propagation requests since the recovery point, also recover the program libraries containing SQL update modules and the libraries containing DBRMs of these SQL update modules to the recovery point. Or, as an alternative, you can use the MVGU RECREATE control statement with the SQLMOD keyword to resynchronize the two libraries with the IMS DPROP directory tables.
- You also need to re-bind the DB2 plans of the propagating application programs to reflect changed DBRMs.
- If the structure of propagated segments has changed or if the RUP is checking DBD version IDs and a DBD version ID has changed, you must recover the IMS DBDLIBs to the same point in time.
- If the structure of propagated segments or of their concatenated keys has changed, you might need to recover IMS databases.
- If the structure of propagated DB2 tables has changed, you might need to either recover the tables or re-extract the IMS data and reload the tables.

Generally, do not propagate affected data during recovery and make sure that you do use the CCU after recovery to check the consistency of IMS and DB2 data.

Chapter 15. Verifying Data Consistency (CCU)

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

This chapter also describes the phases and stages associated with running the CCU. More detailed information on the CCU is in the *Reference*, including sample JCL.

Overview of the CCU

Use the CCU to check consistency between the IMS and DB2 copy of the data.

The CCU:

- Supports only propagation request for generalized mapping cases
- Does not support user mapping (PRTYPE=U)
- Uses the IMS DPROP directory to determine the relationship between IMS segments and DB2 tables
- Uses RUP and HUP (for synchronous) to follow mapping done during propagation
- Compares IMS segments and related DB2 rows for data existence and compares IMS fields and corresponding DB2 columns for data content
- When inconsistencies are detected, generates repair statements for propagated IMS segments (synchronous only) and DB2 tables that are in error.

The CCU runs as a relational application in an IMS batch job step (DBBBATCH or DLIBATCH), or IMS batch message processing region (BMP, IMSBATCH).

The CCU run and repair process is illustrated in Figure 36 on page 220. If the CCU finds inconsistencies, it generates one of the following types of call statement to correct them :

- SQL
- DL/I
- SQL and DL/I

If SQL statements are generated, you can run them through DSNTEP2 or DSNTIAD to restore DB2 tables to the same data content as the IMS database. For synchronous propagation, if DL/I statements are generated, you can run them through DFSDDLT0 to restore the IMS database to the same data content as the DB2 tables.

CCU Execution and Repair Process

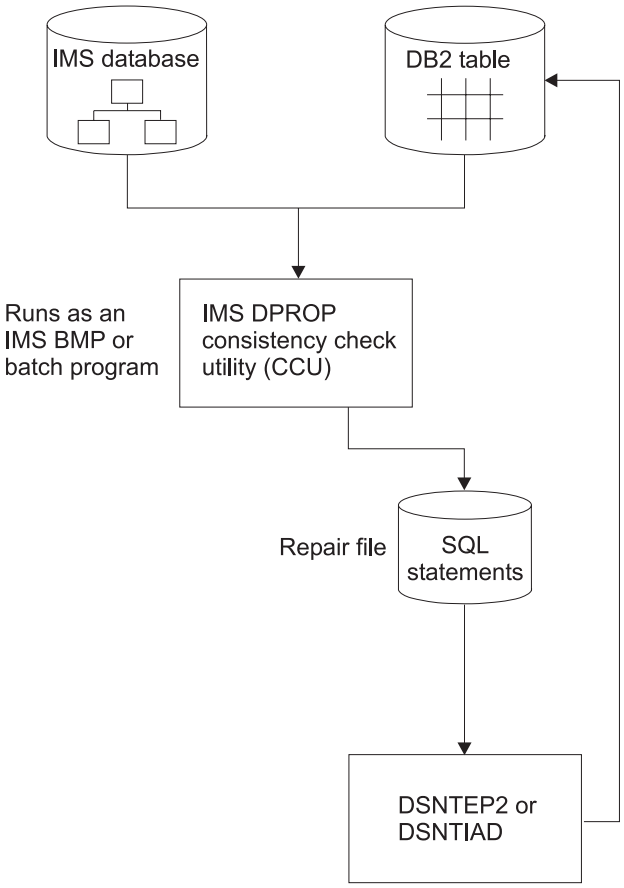


Figure 36. CCU Execution and the Repair Process

The following sections discuss:

- When to use the CCU
- Considerations for LOG Asynchronous propagation, synchronous propagation, and user asynchronous propagation
- Considerations when concurrent updates are being done
- Data availability when the CCU is active
- DB2 referential integrity constraints

When to Use the CCU

You can use the CCU:

- Periodically, to verify the consistency of your propagated data
- As a verification tool for testing propagation requests that you are developing
- After implementing new or changed propagation definitions
- Following a database reload in either IMS or DB2
- After propagation has failed
- After synchronous propagation has been suspended, deactivated, or emergency stopped
- After running database repair programs with PROP OFF control statements

- After encountering operator or application errors, for example, user exit

You might also find other reasons to use the CCU at your installation.

CCU Considerations for Synchronous Propagation

In synchronous propagation, you can run CCU regularly or check data consistency after run after recovery, failure, and extract activities.

Use of the CCU with synchronous propagation is straightforward. There are no special considerations. Refer to the *Reference* for complete information on running 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 222 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 *Reference*. The following sections provide:

- A summary of the phases of the CCU
- CCU verification techniques
- Types of inconsistencies and generated repair statements
- Suggestions for ways to reduce large numbers of inconsistencies
- Some reasons for inconsistencies

Phases of the CCU

The phases of CCU processing are:

Initialization Checks your input control statements, collects IMS DPROP directory information, and builds the mapping control blocks needed by the CCU.

Read and compare Reads and compares both IMS databases and DB2 tables. The CCU completes processing if no inconsistencies are found. Depending on which CCU verification technique you use, the read and compare is done in three phases (hashing technique) or one phase (direct technique). See "CCU Verification Techniques" on page 222.

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.
- For one-way DB2-to-IMS synchronous propagation, the CCU generates DL/I call statements to update the IMS copy and make it consistent with the DB2 copy.
- For two-way synchronous propagation, the CCU generates SQL statements to update the DB2 copy and make it consistent with the IMS copy. The CCU also generates DL/I call statements to update the IMS copy and make it consistent with the DB2 copy.

If the inconsistency is in category

1 The CCU generates an SQL INSERT statement for the missing DB2 row, or a corresponding DL/I DLET call for the IMS segment, or both.

2 The CCU generates an SQL DELETE statement, or a corresponding DL/I ISRT call for the missing IMS segment, or both.

3 The CCU generates an SQL UPDATE statement for the inconsistent columns, or a corresponding DL/I REPL call for the IMS segment, or both.

For every data inconsistency, you must determine whether you want to make data consistent by applying either:

- The generated SQL statement to the DB2 tables using the DSNTEP2 or DSNTIAD programs of DB2
- The generated DL/I call for the IMS segment using the IMS test program DFSDDLT0

The CCU writes the generated repair statements to sequential files that you can edit with TSO/ISPF before passing the files to DSNTEP2, DSNTIAD, DFSDDLT0, or any compatible program you provide.

Generated SQL Repair Statements

SQL statements are created with an asterisk (*) in the first position, so that DSNTEP2 or DSNTIAD treat the statements as comments if the file is accidentally used before proper preparation.

You must decide which portion of the CCU-generated file to use as input to DSNTEP2 or DSNTIAD. Before using the CCU-generated file, sort it with the DFSORT or an equivalent program and do other additional processing as described in the *Reference*.

Generated DL/I Repair Statements

DL/I call statements are created in a format required by DFSDDLT0. The CCU generates the DL/I calls as comments, requiring you to properly prepare the file and verify the generated DL/I call statements with the data copies. Do verification before you run DFSDDLT0. You must decide which part of the generated file to use as input to DFSDDLT0. Before using the generated file, sort it with the DFSORT program or an equivalent program and do other additional processing as described in the *Reference*.

Large Numbers of Inconsistencies

If an unusually large number of inconsistencies occurs, it might be more efficient to re-extract the IMS data and reload the DB2 tables, or to use the DLU, rather than reestablish consistency with CCU-generated repair statements.

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

Some Reasons for Inconsistencies

Some reasons for inconsistencies between propagated IMS and DB2 data are:

- Synchronous propagation is suspended or turned off when the updates are applied. The DB2 tables do not reflect the updates made to the IMS database, or vice versa.
- Propagated tables or databases are updated but not propagated. This occurs, for example, with one-way IMS-to-DB2 propagation if appropriate DB2 security definitions are not in place to prevent users from updating DB2 tables; or it can happen with DB2-to-IMS synchronous propagation if DB2 tables are updated in a non-IMS environment.
- Referential integrity constraints in DB2 do not match the IMS hierarchy when the update is made.
- Faulty database recovery in either IMS or DB2.
  - The recovery are not applied to both copies of the data
  - Erroneous logs or image copies are used in the recovery process, causing the recovered database to be inaccurate
- Either IMS or DB2 has an internal error
- Bad pointers in either an IMS database or DB2 table are encountered

- Mapping during extract and load is not identical to mapping during data propagation.
- Errors occur in the mapping logic of a user-provided IMS DPROP Segment or Field exit routine
- ERROPT=IGNORE is in effect, and errors are encountered
- For LOG Asynchronous propagation, updates are not propagated before the CCU runs
- For user asynchronous propagation, updates are not propagated before the CCU runs.
- For DB2-to-IMS synchronous propagation, monitor class 6 are stopped.

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

Chapter 16. IMS DPROP's 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 problems and resolve them.

You can also use IMS and DB2 when identifying and resolving propagation-related problems:
- IMS DFSERA10 utility, which reads and formats IMS log records
- DB2 DSNILOGP utility, which reads and formats DB2 log records

IMS DPROP Trace Facilities

Use the trace facilities to trace propagation activities of:
- RUP
- HUP
- Selector
- Receiver
- IMS DPROP utilities

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

For synchronous propagation, you can also start tracing using the SCU TRACE ON control statement. Using the SCU TRACE ON control statement allows you trace activities on a system-wide basis; you trace the activities of all jobs doing synchronous propagation.

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 job steps for synchronous propagation, you can write trace output to the IMS log, sequential data set //EKYLOG, and to the sequential output data set //EKYTRACE.

For other types of job steps, for example, IMS DPROP utilities or jobs calling RUP for asynchronous propagation, you can write trace output to the //EKYLOG or //EKYTRACE data set.

If you write the trace to the IMS log or to //EKYLOG, the trace is unformatted and, therefore, requires less CPU overhead, I/O, and external storage. You can also print and format trace records selectively. You can format the trace records with IMS DPROP's EKYZ620X exit routine of the IMS File Select and Formatting Print utility (DFSERA10).

The *Diagnosis* 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
- Status changes made with the SCU
- Program executions in the propagation off (PROPOFF) and propagation suspended (PROPSUSP) modes
- CCU executions
- MVG executions
- Warning messages issued by the MVG
- DLU executions
- Selector executions
- Receiver executions

The following sections cover:

- Using SMF
- Audit extract utility and audit trail table
- Creating an audit trail
- Audit trail table security
- A comparison of audit and trace information
- How the audit trail works with CCU

Using SMF

IMS DPROP writes the records comprising its audit trail to SMF. IMS DPROP uses only one type of SMF record. When you customize IMS DPROP during installation, you define which SMF record type IMS DPROP uses. To start recording IMS DPROP's SMF records, update the SMFPRM*xx* member of SYS1.PARMLIB at your installation. Additional information on this subject is available in the *Installation*.

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 *Reference*.

The audit trail shows:
- When the CCU last ran
- When the data was last known to be consistent
- When data propagation was last suspended or deactivated
- When the RUP or HUP reported a problem
- If a propagation request regenerated

You might want to run the AUDU and load the audit trail table:
- Regularly, whenever SMF data sets are dumped using IFASMFDP.
- Whenever you must diagnose problems. The audit trail table usually contains a combination of historical archived SMF data, and actual current SMF data from the SYS1.MANx data sets.

You will need to set up a procedure for archiving audit trail table data.

Additional information, sample JCL, and control statements for AUDU and the audit trail table are described in *IMS DataPropagator Reference*.

The Figure 37 summarizes the audit trail generation process.

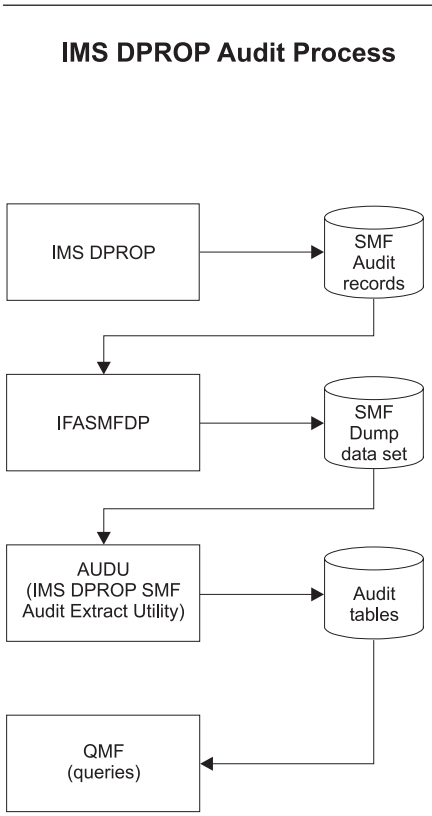


Figure 37. Overview of the IMS DPROP Audit Process

Creating an Audit Trail

The steps in creating an audit trail for IMS DPROP are:

1. Specify the SMF record code to be used by IMS DPROP.

2. Start recording SMF records by updating the SMFPRM*xx* member in SYS1.PARMLIB.
3. Create the audit trail table.
4. Bind a DB2 plan for the AUDU, and grant privileges to run this plan.
5. Grant SELECT privileges on the audit trail table.
6. Develop a procedure for running the AUDU to extract records from SMF and load them into the audit trail table.
7. Implement a maintenance procedure for the audit trail table.

Most of the steps involved in creating an audit trail are performed as part of installation (Step 1 to 5). Steps 3 to 7 do not apply to LOG-ASYNC Selector-only sites that do not have DB2 installed. See *IMS DataPropagator Reference* for details of how to create an audit trail in a Selector site.

Audit Trail Table Security

To access the information stored in the audit trail table, you must grant the SELECT privilege anyone who will use the information. People who SELECT, UPDATE, INSERT, DELETE, and maintain the table (for example, deleting old records) must be granted update privileges.

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

Access to the audit trail table works differently for LOG-ASYNC Selector-only sites that do not have DB2 installed. See *IMS DataPropagator Reference* for details about the audit trail in a LOG-ASYNC Selector-only site.

Comparison of Audit and Trace Information

The audit trail table is a valuable centralized repository of historical information about propagation events. You can use the audit trail table to view information about previous propagation-related events. Audit information differs, however, from trace information. Trace information provides more detailed information.

CCU and the Audit Trail

When writing to the audit trail, the CCU creates at least two records for each phase of processing. Each CCU phase creates a termination record when the phase completes and detail records during execution. Audit trail records are written to SMF for:

- Initialization phase
- IMS read phase for the hashing technique
- DB2 read phase for the hashing technique
- Read phase for the direct technique
- Hash sum compare phase
- Compare phase, when errors are encountered with the hashing technique
- Error location phase
- Final record, written at CCU completion

At completion of its run, the CCU creates a final record for each propagation request. The final record shows inconsistencies between the copies during CCU processing. You can use the information written by the CCU to help resolve inconsistencies between copies of the data.

Monitoring Consistency with the CCU

By monitoring propagated data for consistency between IMS databases and DB2 tables you ensure the usefulness of propagated data. Run the CCU periodically to verify data consistency.

If you find data inconsistencies, check the CCU print output or the audit trail for a description of the inconsistencies. The descriptions contain the identifier of the propagation requests that were checked. With these propagation request identifiers, you can query the message table to determine if any warning messages were written by the MVG when the propagation requests were created. You can then resolve inconsistencies due to propagation failures and erroneous propagation requests.

See Chapter 15, "Verifying Data Consistency (CCU)," on page 219 for a more information on how you can use the CCU.

Monitoring Propagation with the Message Table of the IMS DPROP Directory

The message table contains warning messages issued by the MVG during creation of propagation requests. You can use the message table to analyze propagation failures for a specific propagation request for specific data.

If the MVG does not encounter problems when a propagation request is generated, the message table contains no messages. If warning level diagnostic messages are issued, the message table contains one or more rows that give the propagation request identifier, message number and text issued by MVG, and the names of the database, segment type, and DB2 table. Error level messages cause the generation of a propagation request to fail, and no messages are placed in the message table.

You cannot use the message table for LOG-ASYNC Selector-only sites. Refer to the *Reference* for more information on the message table.

Chapter 17. IMS DPROP Performance and Monitoring

Unlike IMS and DB2, IMS DPROP has few components that you can tune. The best performance environment for propagation is one in which IMS and DB2 are performing at optimum levels. You experience less performance impacts during data propagation when databases and table spaces are well organized. For information about tuning IMS and DB2, see the IMS and DB2 libraries, which contain extensive performance and tuning information.

This chapter discusses:
- IMS DPROP performance
- Monitoring propagation

IMS DPROP Performance

This section describes some aspects of propagation you should consider to maximize performance in your environment. It is organized by propagation phase, with additional performance considerations given at the end of the section. The sections are:

- Mapping and design phase
- Setup phase
- Propagation phase for user asynchronous propagation and synchronous propagation performance
- CCU execution

Mapping and Design Phase

The KEYORDER keyword, used to define propagation requests, can affect performance of the definition process. If you specify ANY, MVG must access the DB2 catalog to determine the proper sequence of the columns of the DB2 primary key. To eliminate this activity, use the proper key sequence (ascending or descending) when you define propagation requests.

In IMS-to-DB2 synchronous propagation, IMS updating applications are impacted by the SQL calls that IMS DPROP issues to propagate changed IMS segments. LOG-ASYNC propagation eliminates the impact on IMS applications.

IMS DPROP minimizes the performance impact of the propagating SQL statements by:

- Using static rather than dynamic SQL statements to propagate IMS database changes to the target DB2 tables.
- Using a WHERE clause on the SQL statements for fields of DB2 indexes. For medium and large tables, this results in a scanning of the index.

For performance reasons, IMS DPROP uses the MVS/ESA VLF to retrieve IMS DPROP control blocks from a data space. VLF greatly reduces the number of SQL calls required to access control information located in the IMS DPROP directory tables. In addition to performance benefits, VLF reduces contention between the RUP and IMS DPROP utilities.

Setup Phase

You can affect the performance of your system by the way you set up IMS DPROP for IMS-to-DB2 or DB2-to-IMS propagation.

IMS-to-DB2 Propagation

The elapsed time required to extract data from an IMS database and load target DB2 tables can be considerable. The elapsed time is directly related to the volume of records being processed. Large buffer pools, cached controllers, and OSAM sequential buffering can reduce elapsed time during the extract. You also have less elapsed time if IMS databases are organized so each database record is placed in as few blocks as possible. Finally, processing the DB2 load in primary key sequence also decreases elapsed time.

You might want to use DataRefresher to *batch* extract requests, allowing multiple segments of the same IMS database be extracted with a single pass through the database.

You also might want to load large DB2 tables using the DB2 LOAD utility. For large tables, using the load utility is generally more efficient than loading the table using SQL insert statements. If you have multiple DB2 tables to load, you might consider loading them in parallel to reduce the elapsed time needed for the extract and load process.

DB2-to-IMS Synchronous Propagation

The elapsed time required for DLU to extract data from DB2 tables and load a target IMS database can be considerable. The elapsed time is directly related to the volume of rows to be processed. Well organized DB2 tables can reduce the time required to extract DB2 data.

In some cases, DLU needs to sort the data, increasing the elapsed time. A sort is required when the:

- DB2 primary sequence of the rows is not identical to the sequence of the IMS keys
- IMS database is an HDAM or DEDB database

If the IMS database is involved in logical relationships, you might need to run the IMS Prefix Resolution and IMS Prefix Update utilities to increase the total elapsed time required to create the IMS database.

For full function IMS databases, you can request that the DLU create a database in the IMS HD unload format. Instead of reloading the unloaded file with the IMS HD Reload utility, you can reload it with the faster Fast Reload utility.

Propagation Phase: Synchronous Propagation Performance

The performance impact for synchronous propagation depends on the direction of propagation:

- IMS-to-DB2
- DB2-to-IMS
- Two-way

IMS-to-DB2 Synchronous Propagation

The performance impact of IMS-to-DB2 synchronous propagation depends largely on the number of IMS update calls to be propagated. In most installations, the number of IMS database reads greatly exceeds the number of updates. Also, most installations propagate only a subset of their IMS databases. You should consider

the number of updating calls against an IMS database when determining if it is a reasonable candidate for propagation to DB2.

The performance impact of synchronous propagation is largely due to the time required by SQL calls updating target DB2 tables. IMS DPROP overhead is small when compared to the overhead required by SQL calls. Therefore, tuning efforts should focus on improving performance of SQL calls.

DB2 RIRs increase the elapsed time and processor time required for data propagation. DB2 indexes can improve the performance of queries, however, might increase the time required for propagation. You should consider performance impact when deciding whether to implement RIRs or DB2 indexes.

After loading the target DB2 tables, you should always run the DB2 RUNSTATS utility to optimize DB2's access to the target table.

In an MPP region, significant CPU time can be spent loading SQL update modules. For performance reasons, RUP maintains in the virtual storage of each propagating region SQL update modules:

- Used during the current application program run
- Associated with RUP PRCBs that are resident in the virtual storage of the propagating region

In addition, RUP maintains the *'n'* most recently used SQL update modules; these are modules used during previous MPP runs. The default for *'n'* is 40. You can change the default with the RESIDENT SQLU control statement specified in the //EKYIN DD data set. Depending on the number of SQL update modules run in an MPP region, you might want to use a lower or higher number than the default.

For better performance in IMS message regions, consider using the MVS/ESA Library Lookaside facility (LLA) to manage libraries containing the SQL update modules generated by MVG. LLA can substantially improve performance, because the SQL updates modules required for a propagation request might be loaded once each time a message processing program is scheduled. To load SQL update modules, combine use of the LLA and the most recently used chain management of RUP.

The performance of IMS-to-DB2 propagation can also be affected by the number of IMS log records resulting from IMS Data Capture.

DB2-to-IMS Synchronous Propagation

The performance impact of DB2-to-IMS synchronous propagation depends largely on the number of SQL updates to be propagated. Consider the number of updated rows when determining if a table is a reasonable candidate for propagation to IMS.

The performance impact of synchronous propagation is largely due to the time required by IMS calls updating target IMS databases. IMS DPROP overhead is small compared to the overhead of updating SQL calls and propagating IMS updates. Therefore, tuning efforts should focus on improving performance of SQL and IMS calls.

The performance of DB2-to-IMS synchronous propagation may also be affected by the number of:

- DB2 log records written by DB2 Data Capture
- IMS log records resulting from propagating IMS updates

In some cases, depending on the buffer space for DB2 logging, HUP's retrieval of the DB2 Data Capture records results in read I/Os for the active DB2 log; infrequently, this might even result in read I/Os for an archived DB2 log.

Two-Way Synchronous Propagation

Use VLF for the IMS DPROP directory tables and status file record. In MPP regions, you can drastically reduce the impact of SQL calls needed to retrieve information from the directory tables. VLF also reduces the possibility of DB2 enqueue conflicts on the tables. Before starting the IMS system, run the SCU with the INIT VLF control statement to preload VLF objects.

In an IMS online environment, you can modify the number of message regions to maintain the same level of transaction throughput after implementing synchronous propagation.

We recommend that you use the following parameters to bind DB2 plans and create DB2 table spaces:

- BIND parameters
  - ACQUIRE(USE)
  - RELEASE(COMMIT)
  - ISOLATION(CS)
  - VALIDATE(BIND)
- Table space definition parameters for both target tables and the IMS DPROP directory tables and their indexes
  - CLOSE(NO), if running with a DB2 release prior to DB2 V2R3
  - Appropriate FREEPAGE specifications
  - Appropriate PCTFREE specifications
  - LOCKSIZE(ANY) or LOCKSIZE(PAGE)

When possible, consider defining your propagating IMS transactions as wait-for-input (WFI). Use of WFI eliminates IMS, DB2, and IMS DPROP path length used for program scheduling and termination activities.

You might want to grant authorization to PUBLIC to run the plans of propagating applications that run in IMS MPP and IFP regions. You can use transaction security for the transaction codes used to call the applications. Granting the plan privilege to PUBLIC can reduce the amount of time required for processing DB2 authorization.

If the DB2 rows and IMS segments are not stored in the same physical sequence, then efficient sequential updates for the source copy results in random, slow propagation of updates. If propagating HDAM or DEDB databases, cluster the DB2 tables in the physical HDAM or DEDB sequence, if practical, to improve performance.

For efficient propagation of sequential updates of HISAM and HIDAM databases, the index for the primary key of the target DB2 tables should be a clustered index. The clustered sequence of the index should be the same as the sequence of the key of IMS root segments.

You might want to redesign current batch programs or re-plan production job streams after implementing synchronous propagation. For critical batch jobs, you can suspend propagation if propagation affects completion of batch runs in the allotted batch window. If you suspend propagation, you usually have to synchronize the data, either by an extract and load process or by applying the missing updates to the target data copy in parallel using your own program.

If you periodically reload propagated IMS databases and the time required to do an extract and load of the DB2 tables is unacceptable, consider reloading the DB2 tables in parallel. Although the load time might increase, you might have a shorter total elapsed time. however, you write an application to perform the DB2 table load.

Sometimes you can moderately decrease CPU time in propagating regions by increasing the number of RUP and HUP PRCBs that IMS DPROP keeps resident in the virtual storage of each propagating MPP region. Keeping RUP PRCBs resident only affects IMS-to-DB2 synchronous propagation in MPP regions. Keeping HUP PRCBs resident only affects DB2-to-IMS synchronous propagation in MPP regions. You can change the number of resident PRCBs with the RESIDENT control statement of the //EKYIN data set.

You might also consider using LLA to manage the library containing IMS DPROP load modules.

Synchronous propagation probably has different impacts on test and production systems. Using separate production and test systems can minimize the impact of updates to the IMS DPROP directory tables. Usually the tables are updated much more frequently in test environments. When the tables are updated, DB2 enqueue conflicts can occur, adversely affecting performance of a production system.

Propagation Phase: User Asynchronous Propagation Performance

In user asynchronous propagation, if you request that the IMS Asynchronous Data Capture exit write changed segments to the log, then your updating applications are impacted by the additional log records. Writing additional log records is usually minor unless your application's performance is constrained by the amount of IMS logging. Examine the various alternatives for externalizing records to determine the best method for your installation. Alternatives might include externalizing:

- The IMS log (OLDS)
- The IMS message queue
- A full function IMS database
- Sequential dependents of a DEDB
- An MVS, or flat, file

SQL call processing affects the receiving program that calls RUP. If you assign control to your own Asynchronous Data Capture exit, your updating applications are impacted by the processing performed in your exit routine. The impact depends as several variables including how many times you update propagated segments and invoke your exit routine.

CCU Execution

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* and *IMS/ESA Administration Guide: Database Manager*. Information on how to run the DB2 Performance Monitor is in *DB2PM Report Reference*.

IMSPARS and IMSASAP II are other IMS monitoring tools you might find useful. Refer to *IMSPARS Program Description and Operation Manual* and *IMSASAP II Program Description and Operation Manual* for more information about these products.

For synchronous propagation, the DB2 EXPLAIN utility can give you information about the access paths selected by propagating SQL calls. It is documented in *DB2 Administration Guide* and *DB2 Utility Guide and Reference*.

The Database Tools product is useful for tuning IMS databases that are involved in data propagation. While no special guidelines can be given for IMS databases

involved in data propagation, a well-tuned and efficient IMS database can be propagated with less performance impact than one that is poorly tuned.

Part 5. Appendixes

Appendix A. JCL Information

This appendix contains detailed information about JCL changes that you can make in the IMS DPROP environment. Some of the modifications are required for particular types of propagation.

- Modifying the archive JCL to create CDCDSs
- JCL changes for synchronous propagation
- JCL changes for DB2

JCL Changes for Synchronous Propagation

This section contains detailed information about JCL changes that you must make in the synchronous propagation environment.

Once you have established a mixed-mode environment for synchronous propagation, you must make additional changes to the IMS JCL so that your IMS DPROP libraries and files can be accessed. You must include the DD statements required by IMS DPROP in the JCL for the propagating IMS batch and dependent regions. In most cases, you need to modify only a few JCL procedures in the IMS procedures library.

Changes for IMS dependent and batch region job streams include the following steps:

- Modify your STEPLIB, JOBLIB, or LINKLIST.

  The SQL update modules, IMS DPROP exits, and most IMS DPROP modules are loaded from STEPLIB, JOBLIB, or LINKLIST. Add the IMS DPROP library that contains these modules to the STEPLIB or JOBLIB concatenations, or to the LINKLIST so that the modules can be found.

  ```
  //STEPLIB   DD  DSN=IMSESA.RESLIB,DISP=SHR
  //          DD  DSN=DPROP.EKYRESLB,DISP=SHR
  //          DD  DSN=USER.LOAD,DISP=SHR
  ```

- Allocate the APF-authorized IMS DPROP load module library to the EKYRESLB DD name.

  Some IMS DPROP modules need to be loaded from an APF-authorized load library. To load these IMS DPROP modules, provide one of the following:

  – Provide an optional EKYRESLB Dynamic Allocation exit routine. The exit routine allocates the IMS DPROP load module library to the EKYRESLB DD name.

  – Provide the EKYRESLB DD statement in the IMS batch and dependent region procedures:

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

  – Allow IMS DPROP to dynamically allocate the data set that you specified during IMS DPROP customization and generation to EKYRESLB.

- Provide an EKYSTATF statement.

  IMS DPROP requires a DD statement that defines the IMS DPROP status file. The data set that is defined in the EKYSTATF DD statement must be specified as a status file during IMS DPROP customization and generation.

  ```
  //EKYSTATF  DD  DSN=DPROP.STATUS.PROD,DISP=SHR
  ```

  Ensure that the EKYSTATF DD statement identifies the status file of the correct IMS DPROP system.

- Optional: Provide an EKYIN DD statement.

  RUP reads control statements from EKYIN; an optional sequential card image data set. This DD statement is defined in the JCL of the IMS batch and dependent regions. It must be a standard MVS sequential file or a member of a partitioned data set with fixed-length records (RECFM=F/FB/FBS, LRECL=80). Here are some samples of the EKYIN DD statement:

  ```
  //EKYIN     DD  *
     or
  //EKYIN     DD  DSN=DPROP.INPUT.PROD,DISP=SHR
  ```

- Optional: Provide an EKYPRINT statement.

  The optional EKYPRINT statement defines a print file that is used to:

  – list the control statements that are contained in the EKYIN sequential card image data set
  – write related error messages

  Here are some samples of the EKYPRINT DD statement:

  ```
  //EKYPRINT  DD  SYSOUT=*
     or
  //EKYPRINT  DD  DSN=DPROP.PRINT,DISP=MOD
  ```

- Optional: Provide an EKYTRACE statement.

  If you use a TRDEST control statement to send trace output to EKYTRACE, provide an EKYTRACE DD statement in the JCL of the propagating IMS batch or dependent regions. Here are some samples of the EKYTRACE DD statement:

  ```
  //EKYTRACE  DD  SYSOUT=*
     or
  //EKYTRACE  DD  DSN=DPROP.TRACE,DISP=MOD
  ```

- Optional: Provide an EKYLOG statement.

  If you use a TRDEST control statement to send trace output to EKYLOG, provide an EKYLOG DD statement in the JCL of the propagating IMS batch or dependent regions. Here is a sample EKYLOG DD statement:

  ```
  //EKYLOG    DD  DSN=DPROP.LOG,DISP=MOD
  ```

- Optional: Provide an EKYSNAP statement.

  If you activate IMS DPROP tracing with a level of 32 (a SNAP dump of the entire MVS task), provide an EKYSNAP DD statement in the IMS batch or dependent region JCL. Here are some samples of the EKYSNAP DD statement:

  ```
  //EKYSNAP   DD  SYSOUT=*
     or
  //EKYSNAP   DD  DSN=DPROP.SNAP,DISP=MOD
  ```

In an IMS message processing region, you can do IMS DPROP initialization multiple times. If you encounter abends or deadlocks, RUP might read, process, and print the control statements many times during the message region run. If you do not allocate EKYPRINT, EKYSNAP, EKYTRACE, or EKYLOG to SYSOUT, specify a disposition of MOD (DISP=MOD) to prevent reuse of the data set to which the messages are being written.

IMS DPROP can generate a large amount of trace data, so define the size for the EKYSNAP, EKYTRACE, and EKYLOG files carefully.

JCL Changes for DB2

When you establish a mixed-mode (IMS to DB2) system for synchronous propagation, you must include certain libraries for DB2 in the JCL of the IMS control, dependent, and batch regions. You must also define the IMS to DB2 connections in subsystem members (SSMs).

For each DB2 connection, the SSMs describe the following elements:

- DB2 subsystem name (SSN)
- language interface tokens (LITs)
- resource translation tables (RTTs)
- command recognition characters (CRCs)
- region error options (REOs)

A propagation environment using IMS DPROP requires IMS-DB2 connections that are just like mixed-mode environments. The following sections describe the JCL changes that are required:

- DB2 JCL changes in the IMS control region
- DB2 JCL changes in IMS dependent regions
- DB2 JCL changes in IMS batch regions
- SSM member in PROCLIB

DB2 JCL Changes in the IMS Control Region

If your IMS system does not already support access to DB2, you must complete the following steps:

1. Establish IMS access to DB2 load modules by adding the DB2 load library to the MVS link list (LNKLST*xx*), or by using a //DFSESL DD statement to concatenate this library with IMS RESLIB. For example:

   ```
   //DFSESL   DD DSN=IMSESA.RESLIB,DISP=SHR
   //         DD DSN=DSN220.DSNLOAD,DISP=SHR
   ```

   If the JOBLIB or STEPLIB data sets are APF-authorized, you can concatenate the DB2 load library with these data sets.
2. Place an SSM member into IMS PROCLIB. The format and content of SSM members are described in "SSM Member in PROCLIB" on page 248.
3. Insert an SSM keyword in the EXEC statement of your IMS control region procedure. The SSM keyword, together with the IMSID name, identifies the SSM member name in IMS PROCLIB.

DB2 JCL Changes in IMS Dependent Regions

The JCL for dependent regions in which propagating IMS applications run must be modified to provide access to DB2. The modifications consist of the following steps:

- Access to the DB2 load modules. Add the DB2 load library to the MVS link list (LNKLSTxx), or use a //DFSESL DD statement to concatenate this library with IMS RESLIB.

  ```
  //DFSESL    DD  DSN=IMSESA.RESLIB,DISP=SHR
  //          DD  DSN=DSN220.DSNLOAD,DISP=SHR
  ```

  If the JOBLIB or STEPLIB data sets are APF-authorized, you can concatenate the DB2 load library with these data sets.
- If the IMSESA.PROCLIB SSM member for the dependent control region is different from the one that is used by the control region, the dependent region JCL must specify the correct SSM member to be used.

For a particular dependent region, IMS DPROP supports propagation to only one DB2 subsystem. Therefore, in the SSM member of a propagating dependent region, you usually define only one connection to a single DB2 system. You might, however, define additional connections to other DB2 systems for use by nonpropagating programs.

If you need to propagate to multiple DB2 systems from one IMS system, you need multiple IMS DPROP systems, with a given IMS dependent region accessing only one IMS DPROP and one DB2 system.

The format and content of SSM members are described in "SSM Member in PROCLIB" on page 248.

DB2 JCL Changes in IMS Batch Regions

For propagating IMS batch regions, change your JCL by completing these steps:

1. If the DB2 load library has not been added to the MVS link list (LNKLSTxx member of SYS1.PARMLIB), concatenate the DB2 load library to the JOBLIB or STEPLIB data sets.
2. Define the IMS to DB2 connection by defining the DB2 connection either in an SSM member or in the //DDITV02 input file.
3. Provide a //DDOTV02 DD statement for a DB2 output file.

Defining the DB2 connection in an SSM member is convenient because you can make all required JCL changes by modifying the DLIBATCH and DBBBATCH procedures (and similar procedures) in the procedure library. Defining the DB2 connection usually does not require that you change the JCL of those propagating jobs that call the DLIBATCH and DBBBATCH procedures. However, there are these two restrictions:

- The name of the DB2 plan must be identical to the name of the IMS application program unless you have defined a DB2 RTT
- The DB2 connection name must be the same as the job name

If these two restrictions are not applicable at your installation, define the DB2 connection in the //DDITV02 file.

Defining the DB2 Connection in an SSM Member

If you want to define the DB2 connection in an SSM member, follow these steps:

- Update the IMS batch JCL procedures (DLIBATCH and DBBBATCH) to include an SSM keyword value, that identifies an SSM member to the batch region. Refer to the IMS Installation Volume 2: System Definition and Tailoring document for a description of the SSM keyword of the DLIBATCH and DBBBATCH JCL procedures.
- Provide an SSM member in the partitioned data set that is referred to by the //PROCLIB DD statement of the batch region. The format and content of SSM members are described in "SSM Member in PROCLIB" on page 248.

If you define the DB2 connection in an SSM member, you must ***not*** provide a //DDITV02 DD statement.

Defining the DB2 Connection in the //DDITV02 File

If you want to define the DB2 connection in the //DDITV02 file, complete the following steps:

- Specify MBR=DSNMTV01 when you invoke the DLIBATCH or DBBBATCH JCL procedure. DSNMTV01 is considered the application program by the DLIBATCH and DBBBATCH JCL procedures. DSNMTV01, in turn, loads the actual application program.
- Provide a //DDITV02 DD statement describing an input file with the DCB attributes of LRECL=80 and RECFM=F or FB. For example:

  ```
  //DDITV02   DD   *
  ```
- Provide in the //DDITV02 input file a record with the following format:

  ```
  ssn,,DSNMIN10,rtt,err,,connection name,plan,prog
  ```

  Where:

  ssn
  is the name of the DB2 subsystem

  DSNMIN10
  must be coded as shown

  rtt
  is the name of an optional resource translation table

  err
  is the region error option — for a propagating batch region, specify *R*

  connection name
  is the name of the DB2 connection— if you do not specify a connection name, the job name is used

  plan
  is the name of the DB2 plan— if you do not specify a plan name, either the name of your IMS program (if no RTT is specified) or the plan name that is associated with the program name by the RTT is used

  prog
  is the name of your IMS application program

Providing a //DDOTV02 DD Statement

Code a DB2 //DDOTV02 DD statement in the batch region JCL. IMS batch support allows the DB2 IMS attachment facility to write the following things to the //DDOTV02 output data set

- messages
- in-doubt SNAP records
- diagnosis records

The JCL that is used to execute propagating IMS batch jobs must include the //DDOTV02 DD statement satisfying these conditions:

- the DCB keyword on this DD statement is mandatory
- RECFM must be V or VB
- LRECL must be at least 3500
- BLKSIZE must be at least the LRECL plus 4

You can code this statement in one of these two ways:

- Allocate //DDOTV02 to a temporary DASD data set, that can be formatted and printed in a subsequent, conditional job step by the IMS File Select and Formatting Print utility (DFSERA10). All messages and diagnostic records are formatted in a meaningful manner. You can add the //DDOTV02 DD and the conditionally executed print job step of DFSERA10 to the JCL procedures used by the propagating batch applications. Here is a sample of the JCL:

```
//DDOTV02   DD  DSN=&TEMP,DISP=(,PASS),
//          SPACE=(TRK,(10,10),RLSE),UNIT=SYSDA,
//          DCB=(RECFM=VB,LRECL=4092,BLKSIZE=4096)
(At the end of the procedure place the following job step)
//*
//*  PRINT DDOTV02 DATASET
//*
//DB2PRINT  EXEC PGM=DFSERA10,COND=EVEN
//STEPLIB   DD   DSN=IMSESA.RESLIB,DISP=SHR
//SYSPRINT  DD   SYSOUT=*
//SYSUT1    DD   DSN=&TEMP,DISP=(OLD,DELETE)
//SYSIN     DD   DSN=SYS1.PROCLIB(DB2PRINT),DISP=SHR
(Add the following member with name DB2PRINT to SYS1.PROCLIB)
CONTROL  CNTL
OPTION   PRINT
```

- Allocate //DDOTV02 to a SYSOUT data set. Although the messages written by the DB2 IMS attachment facility are formatted so that you can retrieve information from them, the in-doubt SNAP and diagnostic records are not formatted. Here is sample JCL to allocate //DDOTV02 to a SYSOUT data set:

```
//DDOTV02   DD  SYSOUT=*,
//          DCB=(RECFM=VB,LRECL=4092,BLKSIZE=4096)
```

SSM Member in PROCLIB

This section describes the SSM members that are used in IMS dependent and batch regions to define the IMS to DB2 connection.

The name of an SSM member is the concatenation of both the of these elements:

- IMS ID (1 to 4 characters) of the IMS online or batch system
- Value of the SSM keyword (1 to 4 characters) provided by the IMS JCL procedure for the dependent or batch region

Each DB2 connection that is defined by an SSM member is described by one record. The SSM member of a batch region contains only one record, since a batch region connects to only one DB2 system. Usually, a propagating dependent region also connects to only one DB2 system. But, if different nonpropagating MPPs that are running in the same message region need to connect to different DB2 systems, then the SSM member contains multiple records.

The format of the record that describes one DB2 connection is shown here:

```
ssn,lit,DSNMIN10,rtt,err,crc
```

Where:

ssn
is the name of the DB2 subsystem

lit is required for dependent regions and is a 4-character alphanumeric field that identifies each DB2 connection— usually, you specify *lit* as SYS1

When an SQL statement is issued, the DB2 IMS attachment facility matches the language interface token (LIT) provided by the language interface module with the *lit* of the SSM member. The DB2 IMS attachment facility does this comparison to determine which DB2 connection processes the SQL statement.

DSNMIN10
must be coded as shown

rtt
is the name of an optional RTT

err
is the region error option— for propagating regions, specify R for the region error option

crc
is a command recognition character used by IMS to identify DB2 commands entered from an IMS terminal using the /SSR command— the default command recognition character is the hyphen (-)

For more detailed information about the SSM member, refer to the*DB2 Administration Guide*.

Appendix B. Language Interface and Multiple DB2 Systems

The SQL statements issued by IMS DPROP in IMS environments are processed through the IMS-provided language interface module DFSLI000. DFSLI000 is usually generated with the language interface token (LIT) of SYS1. The SYS1 LIT is acceptable for both IMS batch regions and dependent regions. For your installation, however, you might want to direct SQL statements of propagating applications to a DB2 system other than the one associated with the SYS1 LIT (for example, if you are accessing multiple DB2 systems from the same IMS dependent region). With synchronous propagation, all propagating applications running in the same IMS dependent region can access only a single DB2 system.

If you are accessing multiple DB2 systems, you need to:

- Define each connection between IMS and DB2 with the SSM member for either the IMS control region or dependent region. Each connection is identified by a different LIT. For more information, refer to DB2 Administration Guide.
- Relink IMS DPROP module EKYY371X. You must:
  - Generate another language interface module with that LIT identifying the DB2 connection to be used by IMS DPROP.
  - Relink IMS DPROP module EKYY371X with the newly generated language interface module. The link must be done into an APF-authorized library.
  - Specify the APF-authorized load library containing the module in the //EKYRESLB DD and //STEPLIB (or //JOBLIB) statements of the IMS dependent region JCL. Your //STEPLIB (or //JOBLIB) libraries do not need to be APF-authorized.

    The name of the EKYY371X IMS DPROP module does not change; therefore, a different load library is required for each copy. If you use an additional library, it should be first in the concatenation specified in the //EKYRESLB DD statement.

The procedure is shown in Figure 38.

```
//         JOB
//*--------------------------------------------------------------------
//*
//* 1ST JOBSTEP: GENERATE/ASSEMBLE A LANGUAGE-INTERFACE MODULE
//*              WITH THE LANGUAGE INTERFACE TOKEN 'SYS2'.
//*
//*--------------------------------------------------------------------
//ASMLI    EXEC  PGM=IEV90,PARM='OBJECT,NODECK'
//SYSLIB   DD   DSY=SYS1.MACLIB,DISP=SHR
//         DD   DSN=IMS310.MACLIB,DISP=SHR
//SYSUT1   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD   SYSOUT=*,DCB=(BLKSIZE=3509)
//SYSLIN   DD   DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(5,5,0)),
               DCB=(BLKSIZE=400),DSN=&&LOADSET
```

Figure 38. Relinking IMS DPROP Module EKYY371X (Part 1 of 2)

```
//SYSIN    DD   *
            TITLE  'IMS SQL LANGUAGE INTERFACE WITH TOKEN ''SYS2'' '
 DFSLI000   DFSLI  LIT=SYS2
            SPACE  2
 DFSLI000   AMODE  31
 DFSLI000   RMODE  ANY
            END
 /*
 //*-------------------------------------------------------------------
 //*
 //* 2ND JOBSTEP: LINK-EDIT THE LANGUAGE-INTERFACE MODULE
 //*              INTO THE DATA-SET DEFINED BY //SYSLMOD DD.
 //*
 //*-------------------------------------------------------------------
 //LINKLI   EXEC  PGM=IEWL,PARM='MAP,LET,LIST,NCAL,RENT'
 //SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE)
 //         DD  DDNAME=SYSIN
 //SYSLMOD  DD  DSN=USER.LOAD(DFSLI002),DISP=SHR
 //SYSPRINT DD  SYSOUT=*,DCB=(RECFM=FB,BLKSIZE=3509)
 //SYSUTI   DD  UNIT=SYSDA,SPACE=(CYL,(3,2))
 //*-------------------------------------------------------------------
 //*
 //* 3RD JOBSTEP: LINK-EDIT THE DPROP MODULE EKYY371X
 //*              WITH THAT LANGUAGE INTERFACE MODULE, WHICH
 //*              HAS JUST BEEN ASSEMBLED/LINKED.
 //*
 //*              INPUT TO THE LINKAGE-EDITOR ARE:
 //*                 1) THE LANGUAGE INTERFACE MODULE WHICH
 //*                    HAS BEEN ASSEMBLED/LINKED IN THE PREVIOUS
 //*                    JOBSTEPS.
 //*                 2) THE LOAD-MODULE EKYY371X, AS SHIPPED BY IBM
 //*
 //*              OUTPUT OF THE LINKAGE-EDITOR IS:
 //*                 1) A NEW COPY OF EKYY371X, WHICH HAS BEEN LINKED
 //*                    WITH THE PREVIOUSLY ASSEMBLED/LINKED
 //*                    LANGUAGE INTERFACE MODULE.
 //*                    NOTE, THE OUTPUT IS STORED INTO ANOTHER
 //*                    LIBRARY THAN THE INPUT-COPY OF EKYY371X.
 //*                    IT IS FROM THIS OTHER LIBRARY, THAT
 //*                    EKYY371X SHOULD BE LOADED BY DPROP; THIS OTHER
 //*                    LIBRARY SHOULD THEREFORE BE DEFINED IN THE
 //*                    //STEPLIB DD AND //EKYRESLB DD STATEMENTS
 //*                    OF THE DEPENDENT IMS REGIONS.
 //*-------------------------------------------------------------------
 //LNKDPROP EXEC PGM=IEWL,PARM='MAP,LET,LIST,NCAL,RENT'
 //SYSLMOD  DD  DSN=USER.LOAD,DISP=SHR
 //LANGINT  DD  DSN=USER.LOAD,DISP=SHR
 //EKYRESLB DD  DSN=DPROP.EKYRESLB,DISP=SHR
 //SYSPRINT DD  SYSOUT=*,DCB=(RECFM=FB,BLKSIZE=3509)
 //SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(3,2))
 //SYSLIN   DD  *
            INCLUDE LANGINT(DFSLI002)
            INCLUDE EKYRESLB(EKYY371X)
            MODE    AMODE(31),RMODE(ANY)
            ENTRY   EKYY371X
            NAME    EKYY371X(R)
 /*
```

Figure 38. Relinking IMS DPROP Module EKYY371X (Part 2 of 2)

Appendix C. Synchronous Propagation Storage Requirements

Since most of the storage areas, control blocks, and modules of IMS DPROP are loaded above the 16MB line, the amount of virtual storage required by IMS DPROP should not be critical in most installations.

This appendix provides information about IMS DPROP's storage requirements in IMS regions doing synchronous propagation.

Virtual Storage Requirements

This section presents:
- Installation-independent requirements
- Installation-sensitive requirements

Installation-Independent Requirements

Part of IMS DPROP's virtual storage requirements is always required and is independent of processing options and workload.

Above the 16MB line, IMS DPROP requires:
- 529 KB for IMS DPROP service load modules
- 16 KB for control blocks that are not in protected storage
- 8 KB for control blocks in protected storage (subpool=230, key=7)

Below the 16MB line, IMS DPROP requires:
- 11 KB for service load modules
- 4 KB for control blocks that are not in protected storage
- 4 KB for control blocks in protected storage (subpool=230, key=7)

Nearly all IMS DPROP service load modules are reentrant and can, therefore, be stored in the link pack area (LPA). Using the LPA for reentrant IMS DPROP service load modules reduces the main storage required by each IMS region.

Installation-Sensitive Requirements

Part of IMS DPROP's virtual storage needs depends on the processing options and workload required. These are:
- To read RUP and HUP PRCBs in storage, IMS DPROP requires an IOAREA. The size of the IOAREA is the size of the largest processed PRCB, rounded up to the next 1 KB boundary.
- IMS DPROP requires 0.6 KB below the line for each DCB used for sequential I/O, such as EKYTRACE or EKYPRINT. This storage is used for the DCB and for a work area, and must be increased by buffers acquired by the access method.
- You can load user-written exit routines either above or below the line. The exact storage requirements varies with your particular implementation.

IMS-to-DB2 Propagation

Within an IMS region doing IMS-to-DB2 propagation, the RUP requires additional virtual storage:
- 1 KB for the RUP main module.

- 12 KB for the RUP work area used for field conversions and other mapping functions. The size of the work area is affected by the number of columns to be propagated and their size. Segment and Field exit routines also influence the size of the area.
- IMS DPROP uses MVS cell pools to store RUP PRCBs in virtual storage. All RUP PRCBs required within one application program run are kept in virtual storage. Also, to reduce path length, IMS DPROP optionally keeps the most recently used RUP PRCBs in virtual storage. IMS DPROP maintains up to the default limit of 20 RUP PRCBs. Or you can specify a value in the RESIDENT RPRCB keyword. Usually, the size of the cell pools is about 200 KB.
- Each SQL update module loaded above the line usually requires between 6 and 12 KB of storage. All of the SQL modules necessary for one application program run are kept in virtual storage. Also, to reduce path length, IMS DPROP optionally keeps the most recently used SQL update modules in virtual storage. IMS DPROP maintains up to the default limit of 40 modules, or you can specify a value in the RESIDENT SQLU keyword.

The RUP main module and the SQL update modules are reentrant and can therefore be stored in the LPA. Using the LPA for these load modules reduces the main storage required by each IMS region.

DB2-to-IMS Synchronous Propagation

Within an IMS region doing DB2-to-IMS synchronous propagation, the HUP requires additional virtual storage:

- 26 KB for the HUP module.
- For each DB2 table that is propagated, HUP keeps the changed data capture table description in storage. The size of the area is 100 bytes for the header and 44 bytes for each column of the table.
- 256 KB for the IFI read buffer.
- Storage for the HUP work areas used for field conversions and other mapping functions. The size of the work area depends on the size of the segment to be propagated and whether all bytes of the segment are propagated. Segment and Field exit routines also influence the size of this area.
- IMS DPROP uses MVS cell pools to store HUP PRCBs in virtual storage. All HUP PRCBs required within one application program run are kept in virtual storage. Also, to reduce path length, IMS DPROP optionally keeps the most recently used HUP PRCBs in virtual storage. IMS DPROP maintains up to the default limit of 20 HUP PRCBs. Or you can specify a value in the RESIDENT HPRCB keyword. Usually, the size of the cell pools is about 200 KB.

The HUP module is reentrant and can, therefore, be stored in the LPA. Using the LPA for this load module reduces the main storage required by each IMS region.

Transient Storage Requirements

Depending on what processing is being done, IMS DPROP might briefly acquire transient storage for writing messages, tracing error processing, and other information. Transient storage requirements can be estimated at 4 to 20 KB per propagating region.

Real Storage Requirements

IMS DPROP does not do page fixing and therefore has no specific real storage requirements. The virtual storage used by IMS DPROP for its load modules, control blocks, and work areas needs to be backed by a reasonable amount of real storage to avoid too much paging and provide reasonable performance.

Appendix D. Converting PRTYPE=F into PRTYPE=E Propagation Requests

In IMS DPROP Version 2, the rules and requirements for PRTYPE=E are very similar to the R1 rules for PRTYPE=Fs. (See “Propagation Requests and Selecting PRTYPEs” on page 11.) PRTYPE=E rules are somewhat more restrictive then the R1 rules for PRTYPE=F so that PRTYPE=E can support DB2-to-IMS propagation.

You can convert a R1 PRTYPE=F into a R2 PRTYPE=E. Conversion is done by changing the PRTYPE parameter and by recreating the propagation request. You might want to convert PRTYPEs if you plan to eventually implement DB2-to-IMS propagation. When converting, note that R2 rules for PRTYPE=E are more restrictive than R1 rules for PRTYPE=F.

The rules for converting a R1 PRTYPE=F into a R2 PRTYPE=E are:

- With a PRTYPE=E, you must not propagate an IMS field to more than one column of the same table if the field is:
  - The IMS key field
  - Part of the IMS key field
  - Mapped to the DB2 primary key

  IMS DPROP does not impose the same restriction on other IMS fields. However, with DB2-to-IMS synchronous propagation, we do not recommend that you propagate the same field to more than one column. Doing so can result in inconsistent data.
- IMS DPROP limits the number of PRTYPE=Es that can propagate a particular segment type.

  With a few exceptions, you cannot create multiple PRTYPE=Es propagating the same segment to and from multiple tables. The exceptions are:
  - IMS segments containing internal segments that are propagated with mapping case 3 propagation requests. Each internal segment can be propagated at most by one PRTYPE=E. The segment containing the internal segment can be propagated by another PRTYPE=E.
  - IMS segments propagated by multiple PRTYPE=E, if all these propagation requests specify a WHERE clause. One segment occurrence should not satisfy the WHERE clause of more then one of these propagation requests. All of the propagation requests must belong to the same mapping case and have the same mapping direction.

  IMS DPROP allows you to propagate the same segment both with PRTYPE=E and with one or multiple PRTYPE=Ls.
- For PRTYPE=E, extension segments of a mapping case 2 propagation request:
  - Must not have an IMS key field
  - Must not have dependent segments propagated by PRTYPE=Es
- For IMS unidirectional logical relationships, the IMS delete rule for the logical parent segment must be PHYSICAL. PRTYPE=F allows both a PHYSICAL and LOGICAL delete rule.
- For paired logical child segment types propagated by PRTYPE=Es, IMS DPROP rejects propagation request definitions if both paired segments are propagated by propagation requests. If the pairing is VIRTUAL, then the physical child

should be propagated. If the pairing is PHYSICAL, then the child with propagated physical dependent segments should be propagated.

- If implementing a DB2 RIR matching an IMS logical parent/child relationship, observe the following rule:

  Match an IMS PHYSICAL delete rule of the logical parent with a DB2 delete rule of ON DELETE RESTRICT PRTYPE=F allows both ON DELETE RESTRICT and ON DELETE CASCADE.
- If performing DB2-to-IMS or two-way synchronous propagation, the physical parent and ancestors of a propagated segment must also be propagated in the same direction.

If you convert a PRTYPE=F into a PRTYPE=E, then you should extend the logic of your Segment and Field exit routines to support DB2-to-IMS mapping, even if you perform only IMS-to-DB2 propagation. Extending the exit routine logic enables CCU to call your exit routines to perform DB2-to-IMS mapping.

Notices

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

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

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

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

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

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

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

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

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

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

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

IBM Corporation

J46/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA
95161-1003
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:

[1] (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. [1] 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
QMF
RACF
SAA
z/OS

Other company, product, and service names may be trademarks or service marks of others.

Glossary of Terms and Abbreviations

A

abort record. An IMS DataPropagator propagation log record (38*nn* or 5938), indicating that the associated unit of work will not be committed by IMS and should not be propagated to DB2. *Compare with commit record*.

ACB. Application control block. Located in IMS.

ACDC. Asynchronous changed data capture.

Apply Program. A component of IMS MQ-DPROP that reads the MQSeries messages containing the changed data and passes it to the RUP. RUP transforms the changed data into relational format and updates the DB2 target tables.

Archive utility. A utility that filters out propagation log records from the records written to the IMS logs and writes them to Changed Data Capture data sets (CDCDSs).

asynchronous changed data capture. An IMS function that captures the changes needed for IMS DPROP asynchronous propagation and saves them on the IMS logs. The function is mandatory for IMS DPROP asynchronous propagation and is either implemented by an SPE (IMS 3.1) or built into the program (subsequent releases of IMS).

asynchronous propagation. The propagation of data at a later time, not within the same unit of work as the update call.

Audit Extract utility. An IMS DPROP utility that inserts the IMS DPROP audit records written to SMF into the IMS DPROP audit table.

AUDU. Audit Extract utility.

B

Batch Log data set. A data set that an IMS batch job uses to store propagation log records needed for IMS DPROP asynchronous propagation.

C

CAF. Call attach facility.

CCU. Consistency Check utility.

CDCDS. Changed Data Capture data sets.

CDCDS Registration utility. An IMS DPROP asynchronous propagation utility that registers new CDCDS to DBRC.

CDCDS Unregistration utility. An IMS DPROP asynchronous propagation utility that deletes CDCDS entries from DBRC.

CDU. CDCDS Unregistration utility.

CEC. central electronics complex.

Changed Data Capture data set (CDCDS). The data sets that the archive utility uses to store the IMS DPROP asynchronous propagation log records filtered during the archive process. CDCDSs contain only the propagation log records. These log records are used by the Selector in place of the corresponding SLDSs, that contain all IMS changes.

Changed Data Capture exit routine. See DB2 Changed Data Capture exit routine

Changed Data Capture function. See DB2 Changed Data Capture function.

commit record. An IMS DPROP asynchronous propagation log record (9928, 37*nn*, 41*nn*, or 5937) indicating that the associated unit of work has been committed by IMS and should be propagated to DB2. *Compare with abort record*.

concatenated key. See "IMS concatenated key" and "conceptual concatenated key."

conceptual concatenated key. The conceptual concatenated key of a segment consists of the concatenated keys of the segment's immediate physical parent and physical ancestors. Unlike the Conceptual *fully* Concatenated key, the conceptual concatenated key does not include the concatenated key of the segment itself.

conceptual fully concatenated key. The conceptual fully concatenated key is an IMS DPROP concept useful for the propagation of entity segments that do not have a unique IMS fully concatenated key; but that are nevertheless uniquely identifiable.

The conceptual fully concatenated key of a segment consists of these parts:
- the concatenated key of the segment
- the concatenated keys of the segment's physical parent and physical ancestors

The conceptual fully concatenated key is therefore the combination of these parts:
- the IMS fully concatenated key

- the ID fields (if any) of the segment that contribute to the concatenated key of the segment
- the ID fields (if any) of the physical parent or ancestors that contribute to the concatenated keys of the physical parent or ancestor

So, the conceptual fully concatenated key is equal to that hypothetical IMS fully concatenated key, that you would see if including the ID fields into the IMS key-field at each hierarchical level.

The concept of conceptual fully concatenated key allows the support of segments with a unique conceptual fully concatenated key, much in the same way as segments with a unique IMS fully concatenated key.

concatenated key. The concatenated key is an IMS DPROP concept useful for the propagation of entity segments that are neither unique under their parent nor have a unique IMS key, but that are nevertheless uniquely identifiable through ID fields.

The concatenated key is a combination of these fields that identify the segment uniquely under its parent:
- the non-unique IMS key field (if any)
- ID fields

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

Consistency Check utility (CCU). An IMS DPROP utility that checks whether the data that has been propagated between IMS and DB2 databases is consistent. If not, it reports the inconsistencies and generates statements the DBA can use to fix the inconsistencies. The CCU is applicable when generalized mapping cases are being used.

containing IMS segment. An IMS segment that contains internal segments (embedded structures) propagated by mapping case 3 Propagation Requests. It is referred to interchangeably as a "containing IMS segment" or "containing segment."

containing segment. See containing IMS segment.

CRU. CDCDS Registration utility.

D

Data Capture exit routine. See IMS data capture exit routine.

data capture function. An IMS function that captures the changes needed for data propagation.

DataRefresher. An IBM licensed program that lets you extract selected operational data on a periodic or one-time basis.

Data Extract Manager (DEM). A DataRefresher component that extracts the IMS data to which changes will subsequently be propagated. DEM also creates control statements for the DB2 Load utility to load the extracted IMS data into DB2 tables.

data propagation. The application of changes to one set of data to the copy of that data in another database system. See also synchronous propagation and IMS DPROP asynchronous propagation.

DataRefresher DEM. DataRefresher data extract manager.

DataRefresher Map Capture exit routine (MCE). See Map Capture exit routine.

DataRefresher UIM. See User Input Manager.

DBRM. Database Request Module.

DB2 commit count. The number of IMS commit records that the IMS DPROP asynchronous propagation receiver is to apply to DB2 before it issues a DB2 commit.

DB2 Changed Data Capture exit routine. The routine to which the DB2 Changed Data Capture function passes the DB2 changes it has captured for propagation. This routine can be the IMS DPROP HUP routine, that propagates data, or your own exit routine.

DB2 Changed Data Capture function. A DB2 function that captures the DB2 changes needed for data propagation.

DB2 Changed Data Capture subexit routine. An optional IMS DPROP exit routine invoked whenever the HUP is called by DB2 changed data capture. The DB2 Changed Data Capture subexit routine can typically be used to perform generalized functions such as auditing all of the captured DB2 changes.

DB2-to-IMS propagation. Propagation of changed DB2 tables to IMS segments. It can be either:
- One-way DB2-to-IMS propagation
- DB2-to-IMS propagation, as part of two-way propagation

DBD. Database definition. The collection of macroparameter statements that describes an IMS database. These statements describe the hierarchical structure, IMS organization, device type, segment length, sequence fields, and alternate search fields. The statements are assembled to produce database description blocks.

DBDLIB. Database definition library.

DBPCB. Database program communication block.

DEDB. Data entry database.

DEM. Data Extract Manager.

directory. See IMS DPROP directory.

DLU. DL/1 Load Utilities. IMS DPROP utilities that are used to create (or re-create) the IMS databases from the content of the propagated DB2 tables. You can use DLU if you have implemented DB2 to IMS or two-way propagation.

DPROP-NR. The abbreviation for IBM IMS DataPropagator MVS/ESA through Version 2.2. At Version 3.1 the product name changed to IMS DataPropagator, abbreviated as IMS DPROP.

E

EKYMQCAP. The Capture component of MQ-DPROP. EKYMQCAP is an IMS data Capture exit routine. It runs as an extension to the updating IMS application programs, but it is transparent to them. EKYMQCAP obtains the changed data from the IMS Data Capture function and sends this data via MQSeries messages to the Apply Program.

EKYRESLB Dynamic Allocation exit routine. An IMS DPROP exit routine that can be used to allocate dynamically the IMS DPROP load module library to the EKYRESLB DD-name.

entity segment. The data being mapped from IMS to DB2 comes from one single hierarchic path down to a particular segment. This segment is called the entity segment. See also mapping case 1.

ER. Extract request.

Event Marker. A component of MQ-DPROP that runs on the same system as the IMS source databases. It is used to identify an event that occurs on the Source System. The customer must execute the Event Marker on the Source System at the time that the event occurs.

The Event Marker transmits an MQSeries message that identifies the event to the Apply Program. This MQSeries message is transmitted in FIFO sequence and in the same Propagation Data Streams as the changed IMS data.

When an occurrence of the Apply Program processes this message, the content of the target DB2 tables of this occurrence of the Apply Program reflect the content of the IMS source databases at the time that the Event Marker was executed on the Source System.

The Event Marker is used for an automated stop of the Apply Program when the content of the target DB2 tables reflects a particular Source System point in time.

exit routines. IMS DPROP contains seven exit routines. See the individual glossary entries for:
- DB2 Changed Data Capture exit routine
- DB2 Changed Data Capture subexit routine
- IMS Data Capture exit routine
- Field exit routine
- Map Capture exit routine
- Propagation exit routine
- Segment exit routine
- User exit routine

extension segment. The data being mapped from IMS to DB2 comes from a single hierarchic path down to an entity segment and from any segments immediately subordinate to the entity segment. The segments subordinate to the entity segment can have zero or one occurrence beneath a single occurrence of the entity segment. This type of subordinate segment is called an extension segment (as it extends the data in the entity segment). See also mapping case 2.

extract request (ER). A DataRefresher request to extract IMS data. Extract requests become IMS DPROP propagation requests once they are validated by the IMS DPROP MCE.

F

Field exit routine. An IMS DPROP exit routine you can write to complement the logic of IMS DPROP's generalized mapping cases. Field exit routines are typically used to convert an individual IMS data field between a customer format IMS DPROP does not support and a format you have defined in your propagation request.

FIFO. First-In-First-Out

fully concatenated key. See IMS fully concatenated key and conceptual fully concatenated key.

G

generalized mapping cases. The mapping cases provided by IMS DPROP. See mapping case 1, mapping case 2 and mapping case 3.

group definition file. The file that the Group Unload utility (GUU) uses to store the IMS sources that it extracts from the IMS DPROP directory tables. *See also, SCF Compare job and SCF Apply job.*

Group Unload utility (GUU). The IMS DPROP asynchronous propagation utility that extracts details of all IMS sources for the specified propagation group from the IMS DPROP directory tables at the receiver site and writes them to the Group Definitions File. *See also, SCF Compare job and SCF Apply job.*

GUU. Group Unload utility.

H

hierarchical update program (HUP). The IMS DPROP component that does the actual DB2-to-IMS propagation. HUP is the IMS DPROP-provided DB2

Changed Data Capture exit routine. The DB2 Changed Data Capture function calls HUP and provides to HUP the changed IMS rows.

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

HUP. Hierarchical Update program.

HSSR. High speed sequential retrieval.

I

ID fields. *Identification (ID) fields* are non-key fields that:
- uniquely identify a segment under its parent
- do not change their value

Typical examples of IMS segments with ID fields, are segments where the data base administrator has not defined the ID fields as part of the IMS Key field. For example because the IMS applications need to retrieve the segment in another sequence than the ascending sequence of the ID fields.

identification fields. See ID fields.

IMS concatenated key. For an IMS segment, the concatenated key consists of:
- The key of the segment's immediate parent, and
- The keys of the segment's ancestors

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

A logical child segment has two concatenated keys: a physical concatenated key and a logical concatenated key. The physical concatenated key consists of the key of the segment's physical parent and the keys of the physical ancestors of the physical parent. The logical concatenated key consists of the key of the segment's logical parent and the keys of the physical ancestors of the logical parent.

IMS Data Capture exit routine. The routine to which the IMS Data Capture function passes the IMS changes it has captured for propagation. For synchronous propagation, this routine can be the IMS DPROP RUP routine, that propagates data, or your own exit routine. For IMS DPROP asynchronous propagation, the data capture exit routine is a program you write that gets the changed data from IMS. Other programs that you write will later invoke IMS DPROP with the changed IMS data.

IMS data capture function. An IMS function that captures the changes needed for data propagation.

IMS DPROP. The abbreviated name for the IBM IMS DataPropagator product. Previously, this product was called IMS DataPropagator, abbreviated as DPROP-NR.

IMS DPROP directory. A set of DB2 tables containing the mapping and control information necessary to perform propagation.

IMS fully concatenated key. For an IMS segment, the fully concatenated key consists of:
- The key of the segment,
- The key of the segment's immediate parent, and
- The keys of the segment's ancestors.

Unlike the IMS concatenated key of the segment, the fully concatenated key includes the key of the segment itself.

IMS INQY data. The first 9904 (update) record in each IMS unit of work (UOW) contains IMS INQY data (transaction name, PSB name, and user ID). This information is written to the PRDS for the propagation group as the first record of the UOW.

IMS log files. The files that IMS uses to store details of all changes to IMS data. See also, batch log data sets, online data sets (OLDSs), system log data sets (SLDSs), and Changed Data Capture data sets (CDCDSs).

IMS logical concatenated key. One of the two IMS concatenated keys of a logical child segment (the other is an IMS physical concatenated key). The logical concatenated key consists of:
- The key of the segment's logical parent, and
- The keys of the physical ancestors of the logical parent.

IMS physical concatenated key. One of the two IMS concatenated keys of a logical child segment (the other is an IMS logical concatenated key). The physical concatenated key consists of:
- The key of the segment's physical parent, and
- The keys of the physical ancestors of the physical parent.

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

internal segments. Internal Segments is the IMS DPROP and DataRefresher term for structures embedded in IMS Segments, that are propagated through mapping case-3 propagation requests. Each embedded structure (i.e. each internal segment), is propagated to a different table; each occurrence of the embedded structure to one row of the table.

invalid unit of work. An IMS UOW that is missing a first record (containing the INQY data). If the IMS DPROP asynchronous propagation Selector detects an

invalid unit, it responds according to what you specified on the INVUOW keyword of the SELECT control statements. If you specified:

IGNORE
The Selector continues processing

STOP The Selector issues an error message and terminates

ISC. Inter-system communications.

ISPF. Interactive system production facility or Interactive structured programming facility.

IXF. Integrated exchange format.

L

LOG-ASYNC. The IMS log-based, asynchronous propagation functions of IMS DPROP.

Once the IMS log records are archived (IMS Online Logs) or de-allocated (IMS Batch Logs) by IMS and then stored in time-stamp sequence, LOG-DPROP reads the IMS logs to find the changed data and then stores the changed data in PRDS datasets. The Receiver component of IMS DPROP reads the PRDSs, transforms the data into the relational format, and applies the changes to the target DB2 tables.

See asynchronous propagation.

logical concatenated key. See IMS logical concatenated key

M

Map Capture exit (MCE) routine. The map capture exit routine provided by DPROP. MCE is used when you provide mapping information through DataRefresher. MCE is called by DataRefresher during mapping and data extract to perform various validation and checking operations. The IMS DPROP MCE should be distinguished from the DataRefresher Map Capture exit, the DataRefresher routine that calls MCE.

mapping case. A definition of how IMS segments are to be mapped to DB2 tables. IMS DPROP distinguishes between mapping case 1, mapping case 2, and user mapping cases.

mapping case 1. One of the generalized mapping cases provided by IMS DPROP. Mapping case 1 maps one single segment type, with the keys of all parents up to the root, to a row in a single DB2 table.

mapping case 2. One of the generalized mapping cases provided by IMS DPROP. Mapping case 2 maps one single segment type, with the keys of all parents up to the root, plus data from one or more immediately subordinate segment types (with a maximum of one occurrence of each segment type per parent), to a row in a single DB2 table.

mapping case 3. One of the generalized mapping cases provided by IMS DPROP. Mapping case 3 supports the propagation of segments containing embedded structures. A typical example of an embedded structure is a repeating group of fields.

- each embedded structure can be propagated to/from a different table. Mapping case 3 propagates each occurrence of an embedded structure, with the key of the IMS segment, and the keys of the physical parent and ancestor, to/from a row of one DB2 table.
- the remaining data of the IMS segment (that is the fields that are not located in a embedded structure) can be propagated to/from another table.

Mapping Verification and Generation (MVG). An IMS DPROP component that validates the mapping information for each propagation request and stores it in the IMS DPROP directory. For a propagation request belonging to a generalized mapping case, MVG generates an SQL update module. MVG is invoked internally by MCE and MVGU.

Mapping Verification and Generation utility (MVGU). An IMS DPROP utility invoked by the DBA. MVGU creates propagation requests when DataRefresher is not used to provide mapping information (i.e., when you put the mapping information directly into the MVG input tables). MVGU also deletes or rebuilds propagation requests in the IMS DPROP directory.

master table. The IMS DPROP directory master table, that is created when IMS DPROP is initialized. It consists of one row, containing system and error information.

MCE. Map Capture exit routine.

MIT. Master Index Table.

MQ-ASYNC. The MQSeries-based, asynchronous propagation functions of IMS DPROP.

An IMS Data Capture Exit routine provided by IMS DPROP obtains the IMS Database changes in real time from IMS and sends the changes via MQSeries messages to an IMS DPROP Apply program. The Apply program reads the MQSeries messages, transforms the data into relational format, and then applies the new data to the target DB2 tables.

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

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

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

MSDB. Main storage database.

MSC. Multisystem communication.

MVG. Mapping Verification and Generation.

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

MVGU. Mapping Verification and Generation utility.

N

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

O

OLDS. Online Data Set.

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

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

P

PCB. Program communication block.

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

physical concatenated key. See IMS physical concatenated key.

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

PR. Propagation request.

PR ID. Propagation request identifier.

PRCT. Propagation Request Control Table

PRDS. Propagation Request Data Set

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

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

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

propagation. See data propagation.

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

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

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

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

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

propagation log records. IMS log records that the IMS DPROP asynchronous propagation Selector writes to PRDSs:
- 9904 (update) records
- Commit or abort records
- SETS/ROLS records

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

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

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

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

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

PRU. PRDS Registration utility.

PSB. Program specification block.

R

RCT. Receiver control table.

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

Applies to LOG-DPROP.

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

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

Applies to LOG-DPROP.

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

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

Applies to LOG-DPROP.

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

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

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

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

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

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

Implementation of RIRs between propagated tables is:

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

RTT. Resource translation table.

RUP. Relational Update program.

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

S

SCF. Selector Control File.

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

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

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

SCU. Status Change utility.

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

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

Applies to LOG-DPROP.

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

Applies to LOG-DPROP.

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

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

Applies to LOG-DPROP.

SLDS. System Log Data Set.

SNAP. system network analysis program

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

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

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

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

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

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

T

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

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

TSMF. Timestamp Marker Facility.

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

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

TW propagation. See two-way propagation.

U

UIM. User Input Manager.

ULR. Uncommitted Log Record.

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

UOW. Unit of work.

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

user exit. See exit routines.

User Input Manager (UIM). A DataRefresher component to which you describe your IMS databases and the mapping between IMS databases and DB2 tables. The mapping is defined by submitting extract requests. You can specify on an extract requests that the UIM is to invoke the DataRefresher Map Capture exit routine provided by IMS DPROP and pass it the DataRefresher mapping definitions of the extract request.

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

V

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

VLF. Virtual Lookaside Facility.

Bibliography

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

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

Other 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 3*

SC26-3265 *DB2 Administration Guide, Version 5*

SC26-4891 *DB2 Command and Utility Reference, Version 3*

SC26-3267 *DB2 Command Reference, Version 5*

SC26-3395 *DB2 Utility Guide and Reference, Version 5*

SC26-4890 *DB2 SQL Reference, Version 3*

SC26-3270 *DB2 SQL Reference, Version 5*

SC26-3269 *DB2 for OS/390 Data Sharing: Planning and Administration, Version 4*

SC26-4248 *DXT Reference*

SC26-4636 *DXT Writing Exit Routines*

SH19-6999 *DataRefresher Command Reference*

SH19-6998 *DataRefresher Exit Routines*

SC26-3066 *IMS/ESA Application Programming: Design Guide, Version 4*

SC26-8016 *IMS/ESA Application Programming: Design Guide, Version 5*

SC26-3062 *IMS/ESA Application Programming: DL/I Calls, Version 4*

SC26-8015 *IMS/ESA Application Programming: Database Manager, Version 5*

SC26-3065 *IMS/ESA Administration Guide: Database Manager, Version 4*

SC26-8012 *IMS/ESA Administration Guide: Database Manager, Version 5*

SC26-3064 *IMS/ESA Customization Guide, Version 4*

SC26-8020 *IMS/ESA Customization Guide, Version 5*

SC26-3076 *IMS/ESA System Definition and Tailoring, Version 4*

SC26-8024 *IMS/ESA Installation Volume 2: System Definition and Tailoring, Version 5*

SC26-4329 *IMS/ESA Utilities Reference: System, Version 4*

SC26-8035 *IMS/ESA Utilities Reference: System, Version 5*

SC26-3072 *IMS/ESA Operations Guide, Version 4*

SC26-8029 *IMS/ESA Operations Guide, Version 5*

GC26-8031 *IMS Release Planning Guide, Version 5*

SC26-3075 *IMS/ESA Administration Guide: System, Version 4*

SC26-8013 *IMS/ESA Administration Guide: System, Version 5*

SC26-4627 *IMS/ESA Utilities Reference: Database, Version 4*

SC26-8034 *IMS/ESA Utilities Reference: Database Manager, Version 5*

GN28-1257	*OS/390 MVS Application Development Guide*
GC28-1473	*OS/390 MVS JCL User\x27–s Guide*
GN28-1427	*OS/390 MVS Initialization and Tuning*
GC28-1449	*OS/390 MVS Setting up a Sysplex*
GC28-1208	*OS/390 Parallel Sysplex Overview: An Introduction to Data Sharing and Parallelism*
GC26-3398	*An Introduction to DataPropagator Relational*
SC26-3399	*IBM DB2 Universal Database Replication Guide and Reference*

Index

Special characters

-DISPLAY DATABASE command 203
-START DATABASE ACCESS(RW) command 215
-START DATABASE ACCESS(UT) command 215
//DDITV02 file 178, 246
//DXTOUT 162
//EKYIN statement 244
//EKYLOG data set 207, 227
//EKYLOG statement 244
//EKYPRINT statement 244
//EKYRESLB statement 243
//EKYSNAP statement 244
//EKYSTATF statement 243
//EKYTRACE data set 207, 227
//EKYTRACE statement 244
//IEFRDER statement 178
//MVGPARM data set 131
/CK field 57
/DBDUMP command 160
/DBDUMP control statement 196
/DBR command 213
/DBR DATABASE command 215
/DBRECOVERY control statement 196
/STA DB command 160
/STO DATABASE command 215
/SX field 57

A

ACQUIRE parameter 156, 236
ACTION parameter 133, 137
ACTIVATE control statement 53, 143, 160, 192, 196
  changing the status of a PR 192
  orderly status changes 196
  privileges required 143
  starting synchronous propagation 160
  use with unqualified table names 53
active state 192
administering plans 158
administrator tasks 3, 7
ALIAS statement 156
ALLOWPROPOFF control statement 193, 203, 215, 216
  executing database repair programs 193
  preventing inadvertent execution of repair programs 216
  required authority 203
  setting propagation mode off 215
ALTER TABLE statement 119, 153, 176
ANY keyword 233
application programs 85, 88
  checkpoint and restart 85
  considerations 85, 88
audit facilities 228
audit trail 95, 229, 230
audit trail *(continued)*
  and the CCU 230
  audit trail records 95
  creating an 229
audit trail table 95, 140, 228, 230
  accessing the 95
  and the AUDU 228
  description 95
  privileges for 140
  security of 230
AUDU (Audit Extract utility) 95, 142, 228
  accessing audit trail 95
  binding plans of 142
  description 228
authorization and privileges 139, 148
authorized state 88
availability of data 86, 221
availability, coordinating IMS and DB2 102
AVU parameter 134

B

backout 86, 87, 178, 212
  dynamic 178, 212
  of committed data 212
  of IMS batch programs 212
  synchronous UOW 86, 87
backup 213
  of databases 213
  of system data sets 213
bad pointers 215
batch backout 212
Batch Backout utility 185, 212
batch programs, granting authorization for 148
batch regions 212, 246
  DB2 JCL changes in 246
  dynamic backout 212
batching extract requests 162
bibliography 273
bidirectional relationships 45
binary integers 80
BIND command 147
BIND COPY command 151
BIND PACKAGE command 52
BIND parameter 136, 236
binding 118, 141, 142, 146, 147, 149, 153, 154, 158
  DB2 plans, re-bind 118
  packages of IMS DPROP modules 141
  packages of SQL update modules 146
  plans of IMS DPROP utilities 142
  plans of propagating applications 147, 149
  plans using DB2 package bind 149
  plans without DB2 package bind 153, 158
binding *(continued)*
  Propagation exit routines 146
  synchronous propagation applications 153
  user asynchronous receiver program 154
BKO=Y keyword 178
BMP (batch message processing) programs 148

C

CAF (call attachment facility) 88
  DB2 functions available with 88
  requests 88
calls 86, 87, 123, 180, 181, 182, 183, 186
  DEF 123
  INIT STATUS GROUPA 86, 180, 181
  INIT STATUS GROUPB 87, 180, 183
  ROLB 87, 182, 186
  ROLS 86
  SETS 86
CANCEL command 136
CASCADE data options 108
CCU (Consistency Check utility) 64, 142, 200, 219, 220, 221, 222, 223, 224, 225, 230, 237
  and RIRs 221
  audit trail considerations 230
  binding plans of 142
  concurrent updates 221
  data availability 221
  description 219, 225
  direct techniques 222
  error location phase 222
  generated repair statements 223
  hashing technique 223
  inconsistencies 223
  initialization phase 222
  performance considerations 237
  read and compare phase 222
  reasons for inconsistencies 224
  running the 222, 225
  suspending synchronous propagation 200
  synchronous propagation 221
  verification techniques 222
  when to use 220
CD keyword 162
cell pools 254
CHANGE.DB command 196
changed data capture 100
character strings 82
CHECK statement 238
checkpoint and restart 85, 211, 212
  frequency of 85
  in IMS and DB2 environment 211
  of an IMS batch program 212
CICS (Customer Information Control System) 101
  environment 101

CLOSE request 88
CLOSE(NO) parameter 236
clustered index 236
collection IDs 146, 149, 150
  defined 149
  granting privileges 146
  using different 150
columns 47, 49, 116, 137, 145
  mapping from fields 49
  PROCSED 137
  propagating a subset of 47
  specifying 116
  updatable 145
commands 52, 116, 122, 123, 124, 131, 136, 143, 147, 160, 162, 163, 196, 203, 213, 215
  -DISPLAY DATABASE 203
  -START DATABASE ACCESS(RW) 215
  -START DATABASE ACCESS(UT) 215
  /DBDUMP 160
  /DBR command 213
  /DBR DATABASE 215
  /STA DB 160
  /STO DATABASE 215
  BIND 116, 147
  BIND PACKAGE 52
  CANCEL 136
  CHANGE.DB 196
  CREATE DATATYPE 122
  CREATE DXTPSB 122
  CREATE DXTVIEW 123
  DISPLAY DATABASE 143
  LIST.DB 203
  SUBMIT 124, 131, 162, 163
COMMIT statement 88
committed data, backout of 212
complementary input data 166
conceptual concatenated key, IMS 61
conceptual fully concatenated key, IMS 60
conceptual key, IMS 60
concurrent updates 221
CONNECT request 88
consistency of data 219, 225
containing segments 26
control blocks 112, 175, 253
  DCB 253
  PCB 112
  RUP PRCB 175
control information 89, 93
control region 245
control statements 53, 93, 136, 137, 143, 205, 209, 237
  ACTIVATE 53, 143
  Also see statements 205
  DEACTIVATE 53, 143
  DELETE 136
  ESTOP 143
  INIT DPROP 143
  INIT STATF 143
  INIT VLF 93
  READOFF 143
  READON 143
  RECREATE 137
  RESET 143
control statements *(continued)*
  RESIDENT 237
  REVALIDATE 137
  SUSPEND 53, 143
conversion 76, 77, 79, 82, 83, 257, 258
  between non-numeric data 82, 83
  between numeric fields 79, 82
  of data 76
  of PRTYPE=F to PRTYPE=E 257, 258
  rules 77
coordinating availability between IMS and DB2 102
COPY statement 113
CPU time limits, increasing 113
CREATE ALIAS statement 155
CREATE DATATYPE command 122
CREATE DXTPSB command 122
CREATE DXTVIEW command 123
CREATE SYNONYM statement 155, 157
CREATE TABLE statement 119, 153, 176
CS (cursor stability) 149
cursor stability (CS) 149
Customer Information Control System 101
  See CICS 101

D

data availability 86
DATA CAPTURE CHANGES option 118, 176
Data Capture exit routine 119
  coexist with another generalized exit 119
data denormalization 31
  Also see data normalization 31
Data Extract Manager 161
  See DEM 161
data normalization 83
data options 107, 109
data resynchronization 214
data sets 99, 124, 131, 207, 213, 227, 229
  //EKYLOG 227
  //EKYTRACE 227
  //MVGPARM 131
  backup and recovery of 213
  EKYLOG 207
  EKYTRACE 207
  FDTLIB 124
  RECON 99
  SYS1.MANx 229
data sharing, intersystem 99
data type support 77
data types 77, 78, 79
  characteristics 77, 79
  date 79
  decimal packed field 77
  decimal zoned field 78
  double-precision floating point number 78
  fixed-length character field 78
  fixed-length graphic field 78
  large integer field 77
  single-byte binary field 77
  single-precision floating point number 78
  small integer field 77
data types *(continued)*
  time 79
  timestamp 79
  variable-length character field 78
  variable-length graphic field 78
Database Recovery Control 99
  See DBRC 99
database request modules 115
  See DBRMs 115
DataRefresher 121, 122, 123, 124, 127, 128, 161, 162, 164
  batching extract requests 162
  commands 122, 123, 124
    CREATE DATATYPE 122
    CREATE DXTPSB 122
    CREATE DXTVIEW 123
    SUBMIT 124
  defining propagation requests using 121, 128
  definition call 123
  EXTLIB 161
  EXTRACT statement 124
  FDTLIB 161
  FDTLIB data set 124
  FIELD statement 123
  MAPEXIT keyword 124
  MAPUPARM keyword 124
  to extract and load data 161, 164
  user mapping cases 127
dates, mapping and conversion between 83
DB repair programs 192
DB2 54, 101
  delete rules 54
  setting up for propagation 101
DB2 Data Capture exit routine 176, 177
  description 176, 177
  environment 176
DB2 monitor trace class 6 119
  See trace class 6 119
DB2 package bind facility 142, 149, 153
  authorization 142
  using 149, 153
DB2 Sign-on Authorization exit 148
DB2-to-IMS propagation 22
  extension segment rules for synchronous 22
  how extension segments propagated 22
DB2-to-IMS synchronous propagation 11, 38, 43, 46, 112, 145, 165, 234, 235, 254
  defining PCBs reserved for HUP 112
  extract and load performance 234
  extract and load, tasks 165
  granting privileges when doing 145
  HERE insert rule 43
  limitations of 43
  of variable-length segments 46
  propagation phase 235
  relational-to-hierarchical, defined 11
  storage requirements 254
  WHERE clause support 38
DB2CDCEX exit routine 119
DB2CDCEX load module 176
DBBBATCH procedure 178
DBCTL support 100

DBD (database description) 105, 106, 111
ACBGEN 105
changing 111
changing, synchronous 105
creating, synchronous 105
EXIT keyword 106, 111
VERSION keyword 111
DBDUMP control statement 196
DBDV keyword 111
DBRC (Database Recovery Control) 99, 193
setting recovery period 99
share control 193
using 99
DBRECOVERY control statement 196
DBRMs (database request modules) 115
DCB (data control block) 253
DDF (Distributed Data Facility) 117, 145
DDITV02 file 246
DEACTIVATE control statement 53, 143, 192, 196, 198
orderly status changes 196
orderly termination of synchronous propagation 198
privileges required 143
switching propagation requests off 192
use with unqualified table names 53
deadlocks 185
DB2 185
IMS 185
DEBUG keyword 227
debug level 227
decimal fields 77, 78, 80
mapping and conversion 80
packed 77
zoned 78
DEDBs (data entry databases) 43, 193
limitations 43
read-only status 193
DEF call 123
default value extension option 22
defining and changing propagation requests 121
defining mapping information 128
definition call 123
DEFVEXT option 22
DELETE control statement (MVGU) 136
delete rules 44, 45, 54, 55
DB2 54
guidelines 44, 45
IMS 54
ON DELETE 55
DELETE statement (SQL) 136
deleting a propagation request 136
DELEXT parameter 135
DEM (Data Extract Manager) 159, 161
denormalizing data, performance 31, 32, 83
DENYPROPOFF control statement 203, 215
required authority 203
dependent regions 245
design considerations 11
DFSBBO00 212
DFSDDLT0 program 215, 219, 224
DFSERA10 utility 208, 228
DFSLI000 module 251
DFSORT program 224
DFSURGU0 utility 166, 214
diagnosis tools 227, 231
direct technique, CCU 64, 222
directory ID 92
directory, IMS DPROP 89, 92
DISCONNECT request 88
DISPLAY DATABASE command 143
DISPLAY statement 203
displaying system information 203
Distributed Data Facility (DDF) 117, 145
DL/I (data language I) 224
DL/I Test program 215
DLIBATCH procedure 178
DLU (DL/I Load utilities) 30, 142, 165, 166, 167, 168, 171, 172
binding plans of 142
complementary data 167
description 165, 172
input and output 166
input selection rules 167
logical relations, paired segments 171
operating considerations 168
processing of internal segments 30
restrictions 166
double-precision floating point number 78
DPRIFLD table 128, 129
DPRIPR table 128, 131
DPRISEG table 128, 129
DPRITAB table 128, 129
DPRIWHR table 128
DPROP SUPPORT parameter 101, 114, 145, 202
DSNMAPN macro 158
DSNTEP2 program 214, 219, 224
DSNTIAD program 214, 219, 224
DUBIOUS keyword 201
DXT xii
terminology xii
DXTPCB 162
DXTVIEW 162
dynamic backout 178, 212

E

EDEACTIVATE control statement 192, 196, 199
emergency termination of synchronous propagation 199
inconsistencies resulting from 196
switching propagation requests off 192
EKYGSYS macro 111
EKYLOG data set 227
EKYMCE00 exit routine 124, 161
EKYRUP00 keyword 175
EKYTRACE data set 227
EKYY371X module 251
EKYZ620X exit routine 208, 228
embedded structures 23
emergency stopped state 191, 202
ENABLE parameter 156
ENFORCE NO 163
enqueue conflicts, DB2 237
entity segments 19, 20
environments 88, 95, 99, 100, 101, 175, 176
CICS 101
DBCTL 100, 101
HUP 176
IMS 99, 101
IMS DPROP 95
mixed-mode 88
multiple IMS DPROP systems 95
production 95
RUP 175
test 95
XRF 100
ER 162
See extract requests 162
ERRCTL control statement 189, 204
ERROPT 102, 133, 178, 179, 188, 204
BACKOUT parameter 178
control statement 204
IGNORE parameter 102, 179, 188
options 133
error 177, 179, 188, 190
codes 179
See SQL error codes 179
IMS 188
in synchronous mode 177, 190
messages 188
Also see messages 188
limiting 188
error location phase 222
error options, changing 204
ESTOP control statement 143, 165, 202
examples 51, 65, 67, 71, 109, 111
EXIT keyword 109, 111
mapping keys PRTYPE=E 65, 67
mapping keys PRTYPE=L 71
using propagation request set 51
EXCLUDE control statement 169
EXIT keyword 106, 175
synchronous propagation 106, 175
exit routines 28, 117, 119, 124, 148, 161, 228
DB2 Sign-on Authorization 148
DB2CDCEX 119
DPROPNR MCE 161
EKYMCE00 124, 161
EKYZ620X 228
Validation 117
EXITNAME parameter 135
EXPLAIN utility 238
extended function PRs 62
See PRTYPE=E 62
Extended Recovery Facility 100
See XRF 100
extension segments 20, 21
defined 20
rules for mapping fields in 21
EXTLIB 161
extract and load phase 234
performance 234
extract requests (ER) 121, 162
batching 162
relationship to propagation requests 121
EXTRACT statement 124
extracting data 159, 161, 163, 164, 165

extracting data *(continued)*
for DB2-to-IMS synchronous propagation 165
for IMS-to-DB2 propagation 159, 164
with DataRefresher 161, 164
with your programs 163

F

failure 53, 86, 87, 177, 183
referential integrity relationship (RIR) 53
synchronous propagation 86, 87, 177, 183
Fast Path 148
See DEDBs 148
Fast Reload utility 234
FDTLIB 124, 161
field formats supported 74, 83
FIELD statement 59, 123
fields 47, 49, 57, 59, 75, 77, 78, 79, 80, 83, 117, 129, 180
/CK 57
/SX 57
a subset of the fields in a segment, asynchronous 47
decimal packed 77
decimal zoned 78
decimal, mapping and conversion 80
describing 75
fixed-length character 78
fixed-length graphic 78
ID 59
large integer 77
mapping and conversion between 79, 83
mapping to columns 49
propagating a subset of 47
RVALCSTC 117
single-byte binary 77
small integer 77
specifying those to be propagated 129
SQLERRMC 180
variable-length character 78
variable-length graphic 78
File Select and Formatting Print utility 208, 228
fixed-length character field 78
fixed-length graphic field 78
floating point numbers 82
foreign key, DB2 61
formatting and printing routine 208
FREEPAGE parameter 236
full function PR 12, 18
See PRTYPE=F 12
fully concatenated key, IMS 59
functional identifiers 148

G

generalized mapping 18, 30, 41, 176, 177
See also mapping case 1, mapping case 2, and mapping case 3
and HUP processing 177
and RUP processing 176
generalized mapping *(continued)*
selecting a mapping case 18, 30
selecting mapping options 30, 41
global master timestamp (GMTS) 93
GN call 182
granting authority and privileges 139, 148
graphic strings 82
GU call 182
guidelines 43, 45, 53
for DB2 referential integrity 53
for IMS 45

H

hashing technique, CCU 223
HASHONLY keyword 238
HD Reload utility 234
HD Reorganization Unload utility 214
hierarchical-to-relational propagation 11
HR (hierarchical-to-relational) propagation 11
See IMS-to-DB2 propagation
HSSR DB Unload utilities 166
HUP (hierarchical update program) 112, 176, 177, 180, 190, 205, 209
alias DB2CDCEX 176
control statements 205, 209
defining PCBs for 112
environment 176
error handling 177, 190
normal processing 176
support for INIT STATUS calls 180
HUP PCBs 112

I

ID fields, mapping 35, 59
IDs (identifiers) 50, 92, 111, 124, 129, 148, 149, 150
collection 149, 150
directory 92
functional 148
PR 124, 129
PRSET 50
version 111
IEFRDER statement 178
IFASMFDP 229
IFI COMMAND, DB2 119, 176
IFP (IMS Fast Path) region 148
Image Copy utility 214
implementing propagation 138
IMS 54, 99
delete rules 54
setting up for propagation 99
IMS Asynchronous Data Capture function 106
EXIT keyword 106
IMS Data Capture exit routine 106, 175, 176
EXIT keyword 106
IBM-supplied 175, 176
description 175, 176
IMS DPROP directory 89, 92, 140, 217, 231
granting privileges for 140
IMS DPROP directory *(continued)*
monitoring propagation with 231
recovery of 217
IMS DPROP Map Capture exit 161
See MCE 161
IMS DPROP phases 3, 7
administrator tasks 3, 7
IMS DPROP status file 205
IMS-to-DB2 propagation 45, 106, 118, 159, 234, 253
binding DB2 plan again 118
EXIT keyword 106
extract and load performance 234
extract and load, tasks 159
of variable-length segments 45
performance, synchronous 234
storage requirements 253
IMS-TO-DB2 propagation 164
extract and load, tasks 164
IMSASAP II 238
IMSPARS 238
inactive state 192
increasing CPU time limits 113
indexes 57, 58, 236
clustered 236
truly unique 58
unique 57, 58
INIT DPROP control statement 143
INIT STATF control statement 143, 205
INIT STATUS GROUPA call 86, 180, 181
INIT STATUS GROUPB call 87, 180, 183
INIT VLF control statement 93, 236
initial bind 115, 153
initial load 217
initialization phase 222
initializing 93, 204, 205
IMS DPROP system 204
VLF objects 93, 205
insert operations in load mode 100
internal segments 26, 28, 30
defined 26
description 28, 30
intersystem data sharing 99
IOAREA 253
ISOLATION(CS) parameter 156, 236

J

JCL 94, 126, 150, 152, 154, 162, 243, 244, 245, 248
//DXTOUT 162
changes for DB2 245
changes for IMS DPROP 243
changes for Sysplex 94
DataRefresher UIM 126
EKYIN statement 244
EKYLOG statement 244
EKYPRINT statement 244
EKYRESLB statement 243
EKYSNAP statement 244
EKYSTATF statement 243
EKYTRACE statement 244
for binding DB2 packages 150
for binding DB2 plans with bind package 152
for binding DB2 plans without DB2 package bind 154

JCL *(continued)*
for propagating IMS batch jobs 248
job control language 162
See JCL 162

K

key field, IMS 59
key mapping rules 59, 74
KEYONLY keyword 238
KEYORDER keyword 135, 233
keys 45, 59, 60, 61, 74
conceptual 60
conceptual concatenated 61
DB2 foreign 61
DB2 primary 45, 61
definitions 59, 61, 74
ID fields 59
IMS conceptual fully concatenated 60
IMS fully concatenated 59
IMS key field 59
IMS logical concatenated 59
IMS physical concatenated 59
mapping rules 59, 61, 74
NAME 59
keywords 30, 51, 52, 59, 106, 111, 112, 113, 116, 124, 162, 163, 175, 178, 189, 201, 204, 213, 227, 233, 238, 254
ANY 233
BKO=Y 178
CD 162
DBDV 111
DEBUG 227
DUBIOUS 201
EKYRUP00 175
EXIT 106
HASHONLY 238
KEYONLY 238
KEYORDER 233
LIST 112
MAPEXIT 124
MAPUPARM 124
MAXPR 189, 204
MAXSSAUD 189, 204
MAXSSWTO 189, 204
MEMBER 116
NEWCYCLE 204
NOFEOV 213
PCBNAME 112
PKLIST 116
PROCLIM 113
PRSET 51
QUALIFIER 52
RESIDENT HPRCB 254
RESIDENT RPRCB 254
RESIDENT SQLU 254
SEG 30
UNLIMITED 204
USERDECK 163
VERSION 111

L

language interface module 251
language interface token (LIT) 251
large integer field 77
Library Lookaside facility (LLA) 235
limited function PRs 69
See PRTYPE=L 69
LIST keyword 112
LIST.DB command 203
LIT (language interface token) 251
LLA (Library Lookaside facility) 235
load mode 100
Load utility 217, 234
loading data 159, 161, 163, 164, 165, 216
database reorganizations 216
for DB2-to-IMS synchronous propagation 165
for IMS-to-DB2 propagation 159, 164
with DataRefresher 161, 164
with your programs 163
LOCKSIZE(ANY) parameter 236
LOCKSIZE(PAGE) parameter 236
LOG-ASYNC propagation 101
in a CICS environment 101
logical child segments 107
logical children 44
logical concatenated key 59
LOGICAL delete rule 55, 257
logical parents 41, 44
and delete rule 44
propagating with a WHERE clause 41
logical relationships 44, 45
rules 44, 45

M

macros 106, 111, 113, 117, 158, 183
DSNDRVAL 117
DSNMAPN 158
EKYGSYS 111
SEGM 106
TRANSACT 113, 183
maintenance and control phase 3
administrator tasks 3
maintenance, database 211, 218
Map Capture exit routine 161
See MCE 161
MAPCASE parameter 132
MAPDIR parameter 11, 133
MAPEXIT keyword 124
mapping 11, 30, 41, 43, 49, 58, 61, 74, 77, 79, 82, 83
between fields and columns 49
between non-numeric data 82, 83
between numeric fields 79, 82
conversion rules 77
design considerations 11
key mapping rules 61, 74
options 30, 41
description 30, 41
rules, restrictions, guidelines 43, 58, 74, 83
mapping and design phase 3, 233
administrator tasks 3
performance 233
mapping case 1 19, 20
mapping case 2 20, 23
mapping case 3 23, 30
mapping cases 18, 30
mapping cases *(continued)*
selecting 18, 30
MAPUPARM keyword 124
MAPUPARM parameter 131
matching RIRs 54
MAXERROR parameter 133
MAXPR keyword 189, 204
MAXSSAUD keyword 189, 204
MAXSSWTO keyword 189, 204
MCE (Map Capture exit) 121, 161
ER validation 121
relationship to DataRefresher 161
MEMBER keyword 116
Message Processing Facility, MVS 189
message table 231
messages 189
See also error messages
suppressing 189
mixed-mode 85, 88, 114
converting to 114
environment 88
processing 85
MODE=MULT parameter 183
MODE=SNGL parameter 183
model HUP PCB 112
modes of synchronous propagation 191, 209
modules 176, 208, 251, 253
DB2CDCEX 176
DFSLI000 251
EKYY371X 251
language interface 251
resident SQL update 208
storage requirements 253
monitor trace class 6 119
See trace class 6 119
Monitor utility 238
monitoring propagation 238
MPPs (message processing programs) 148
multiple DB2 systems 251
multiple IMS DPROP systems 95
MVG input tables 95, 128, 131, 140
defining propagation requests using 128, 131
description 95
privileges for 140
MVGU (Mapping Verification and Generation utility) 129, 136, 138, 142
binding plans of 142
control statements 136, 138
executing 129
MVS 189, 254
cell pools 254
Message Processing Facility 189

N

NEWCYCLE keyword 204
NOCASCADE data options 108
NODATA suboption 109
NOFEOV keyword 213
non-authorized state 88
non-matching RIRs 57
NOPATH data option 107
NOPATH suboption 108
normalizing data 83

not emergency stopped state 191

O

ON DELETE delete rule 55, 221, 258
  CASCADE 55, 258
  RESTRICT 55, 221, 258
  SET NULLS 55
one-way DB2-to-IMS synchronous propagation 56, 58, 117
  and truly unique secondary indexes 58
  defining RIRs for 56
  protecting tables from updates 117
one-way IMS-to-DB2 propagation 56, 57, 58, 117, 144
  defining RIRs for 56
  granting privileges when doing 144
  implementing non-matching RIRs for 57
  protecting tables from updates 117
  unique DB2 indexes and 58
OPEN request 88
operating environment 95
  See environments or operations 95
operations 102
  reducing risk with ERROPT=IGNORE 102
orderly status changes 196
OWNER parameter 151

P

package bind function 115
  not using 115
  using 115
package collection 149
packages 141, 146, 149
  binding 141, 146
  described 149
paired segment types 171
parameters 11, 32, 35, 37, 59, 101, 114, 126, 131, 132, 133, 134, 135, 136, 137, 145, 151, 152, 156, 175, 176, 178, 183, 202, 236
  ACQUIRE(USE) 156, 236
  ACTION 133, 137
  AVU 134
  BIND 136, 236
  CLOSE(NO) 236
  DATA CAPTURE CHANGES 176
  DELEXT 135
  DPROP SUPPORT 101, 114, 145, 202
  ENABLE 156
  ERROPT 133, 178
  EXIT 175
  EXITNAME 135
  FREEPAGE 236
  ISOLATION(CS) 156, 236
  KEYORDER 135
  LOCKSIZE(ANY) 236
  LOCKSIZE(PAGE) 236
  MAPCASE 132
  MAPDIR 11, 133
  MAPUPARM 131
  MAXERROR 133
parameters *(continued)*
  MODE=MULT 183
  MODE=SNGL 183
  OWNER 151
  PATH=DENORM 32, 35
  PATH=ID 35, 37
  PCBLABEL 136
  PCTFREE 236
  PERFORM 135
  PERFORM(BUILDONLY) 126, 137
  PKLIST 152
  propagation 131, 136
  PROPSEGM 135
  PROPSUP 134
  PRSET 134
  PRTYPE 132
  QUALIFIER 151
  RELEASE(COMMIT) 236
  SEQ 59
  TABQUAL2 133
  VALIDATE(BIND) 151, 156, 236
parent segments 40
PATH data 30, 37, 107
  description 30, 37
  PATH or NOPATH option 107
PATH suboption 108
PATH=DENORM parameter 32, 35
PATH=ID parameter 35, 37
PCB (program control block) 112
  HUP 112
PCBLABEL parameter 136
PCBNAME keyword 112
PCTFREE parameter 236
PERFORM parameter 135
PERFORM(BUILDONLY) parameter 126, 137
performance 32, 233, 234, 237, 238
  description 233, 238
  extract and load 234
  improving by denormalizing 32
  mapping and design phase 233
  propagation phase, synchronous 234
  propagation phase, user asynchronous 237
  setup phase 234
  synchronous propagation 233
Performance Monitor utility 238
phases of propagation 3
  administrator tasks 3
phases, CCU read and compare 222
physical concatenated key 59
PHYSICAL delete rule 55
physically paired segment types 44, 171
PKLIST keyword 116, 152
  bind, synchronous 116
  binding DB2 plans 152
plans 115, 118, 142, 147, 158
  administering 158
  binding 142, 147
  binding again 118
  initial bind 115
PR (propagation request) 18, 19, 50, 51, 137, 191, 195, 197, 198, 199, 200
  activating 197
  controlling with SCU 195, 200
  deactivating 198
  defining 19, 51
PR (propagation request) *(continued)*
  with qualified table names 51
  description 18
  emergency deactivating 198
  revalidating 137
  sets 50
  suspending 199
  synchronous propagation states 191
  using propagation request sets 50
PR ID 124, 129
PRCB (propagation request control block) 175, 208
Prefix Resolution and Update utility 234
primary key columns 116
primary key, DB2 45, 61
  for IMS 45
  for propagation 45
privileges 139, 140, 146, 148
  for audit trail table 140
  for IMS DPROP directory tables 140
  for MVG input tables 140
  for propagating collections 146
  granting 139, 148
problem determination tools 227, 228, 231
  audit facilities 228
  CCU 231
  description 227, 231
  message table 231
  SMF 228
  trace facilities 227
processing 85, 175, 176, 183
  HUP 176
  mixed-mode 85
  RUP 175
  RUP and HUP error 183
processing option 112
PROCLIB 248
PROCLIM keyword 113
PROCOPT=A 112
PROCOPT=L or LS 100, 217
PROCSED column 137
production environment 95
programs 114, 163, 192, 214, 215, 219, 224
  DB repair 192
  DFSDDLT0 215, 219, 224
  DFSORT 224
  DL/I Test 215
  DSNTEP2 214, 219, 224
  DSNTIAD 214, 219, 224
  extracting and loading data with your 163
  mixed-mode 114
PROP LOAD control statement 100, 206, 217
PROP OFF control statement 175, 206, 215, 216
PROP OFF mode 192
PROP SUSP control statement 176, 188, 192, 206
propagating 45, 47, 48, 50, 74, 83
  a subset of columns in a table 48
  a subset of the fields in a segment, synchronous 47
  multiple propagation requests to the same table 50

propagating *(continued)*
one segment to or from multiple tables 50
rules, restrictions, guidelines 74, 83
using propagation request sets 50
variable-length segments 45
propagation 11, 43, 58, 131, 136, 146, 185, 206
controlling 206
design considerations 11
direction 11
emergency stopped or deactivated 185
granting privileges for propagating collections 146
parameters 131, 136
rules, restrictions, guidelines 43, 58
propagation phase 3, 234, 237
administrator tasks 3
synchronous propagation performance 234
user asynchronous propagation performance 237
propagation request (PR) 11, 52, 121, 131, 136, 137
defining 52, 121, 136
with unqualified table names 52
deleting 136
description 11
parameters 131, 136
rebuilding 137
replacing 137
PROPDLOK 182
PROPOTHR 181
PROPSEGM parameter 135
PROPSUP parameter 134
PROPUNAV 180
PRSET ID 50
PRSET keyword 51, 134
PRTYPE=E 12, 15, 22, 29, 38, 54, 62, 69, 257, 258
and internal segments 29
and RIRs 54
converting to 257, 258
described 12, 15
extension segment rules for 22
key mapping rules 62, 69
WHERE clause support 38
PRTYPE=F 12, 18, 257, 258
converting to PRTYPE=E 257, 258
described 12, 18
PRTYPE=L 12, 16, 54, 69, 73
and RIRs 54
described 12, 16
key mapping rules 69, 73
PRTYPE=U 12, 17
described 12, 17
PRTYPEs 11, 18, 59, 74, 132
key mapping rules 59, 74
parameter 132
selecting 11, 18
PSW key 8 88

Q

qualified table names 51
QUALIFIER keyword 52, 151
Quiesce utility 213

R

read and compare phase 222
read-only 141, 193, 194
access mode (DB2) 194
IMS DPROP collection 141
status (IMS) 193
read-write IMS DPROP collection 141
READOFF control statement 143, 200, 201, 214
controlling DB2 201
controlling IMS 200
resynchronization 214
READOFF DB2DB control statement 215
READON control statement 143, 200, 201, 214
controlling DB2 201
controlling IMS 200
resynchronization 214
READON DB2DB control statement 215
real storage requirements 255
rebuilding a propagation request 137
receiver plans 147, 149, 153
binding privilege 147
binding with DB2 package bind 149
binding without DB2 package bind 153
recommendations 13, 40, 41
for propagating logical parents with WHERE clause 41
for propagating parent segments with WHERE clause 40
for selecting propagation request types 13
RECON data set 99
recovery 213, 214, 217
IMS DPROP directory 217
of databases 213
of system data sets 213
point-in-time 214
timestamp 214
recovery period, setting 99
RECREATE control statement 137
recreating VLF objects 93
refreshing VLF objects 93
region error option (REO) 178
registering 99
databases in DBRC 99
RELEASE(COMMIT) parameter 236
reload and extract 236
remote updates, SQL 101
REO (region error option) 178
reorganization and database load 216
repair functions, repair functions 215
repair programs 192, 215, 216
and PROP OFF mode 192
preventing inadvertent execution of 216
user-written 215
repair statements 223, 224
Repair utility 215
replacing a propagation request 137
requests, CAF 88
requirements 39, 45
for DB2 primary key 45
requirements *(continued)*
for PRTYPE=E with WHERE clause 39
RESET control statement 143, 202
RESIDENT control statement 209, 237
RESIDENT HPRCB keyword 254
resident PRCBs 209
RESIDENT RPRCB keyword 254
resident SQL update modules 208
RESIDENT SQLU control statement 209, 235
RESIDENT SQLU keyword 254
resource translation table (RTT) 158
resources, unavailable 186, 187
restart 85, 211, 212
in IMS and DB2 environment 211
of an IMS batch program 212
of IMS online and DB2 212
restrictions 166
DLU 166
resynchronization of data 214
REVALIDATE control statement 137
RH (relational-to-hierarchical) synchronous propagation 11
See DB2-to-IMS synchronous propagation
RIRs (referential integrity relationships) 53, 54, 56, 57, 221
and the CCU 221
DB2 delete rules for matching 54
defining RIRs for 57
defining to match IMS relationships 54
for one-way DB2-to-IMS synchronous propagation 56
for one-way IMS-to-DB2 propagation 56
guidelines for 53, 57
matching 54
non-matching 57
rules 53
ROLB call 87, 182, 186
ROLLBACK statement 88
ROLS call 86
RTT (resource translation table) 158
rules 21, 33, 36, 44, 45, 54, 59, 61, 74, 77, 195, 221, 257
for conversion 77
for converting PRTYPE=F 257
for DB2 delete 54
for IMS delete 54
for logical relationships 44, 45
for mapping fields in extension segments 21
for PATH=DENORM 33
for PATH=ID 36
key mapping 59, 61, 74
ON DELETE 195
ON DELETE RESTRICT 221
rules, restrictions, guidelines 43, 58, 74, 83
RUNSTATS utility 235
RUP (relational update program) 175, 177, 180, 190, 205, 209
control statements 205, 209
environment 175
error handling 177, 190

RUP (relational update program) *(continued)*
normal processing 175
support for INIT STATUS calls 180
RUP PRCB 175

S

scenarios 96, 169, 195
for one or multiple IMS DPROP 96
setting propagation request state, synchronous 195
using DLU, complex 169
using DLU, simple 169
SCU (Status Change utility) 52, 142, 160, 178, 179, 189, 194, 195, 200, 201, 202, 205, 216
alternative to using 160
binding plans of 142
controlling DB2 databases and table spaces 201
controlling full-function IMS databases 200
controlling propagation requests 195
controlling the IMS DPROP system 201
description 194, 205
ERRCTL control statement 189
ERROPT=BACKOUT 178
ERROPT=IGNORE 179
extracting and loading data 160
general service functions 202
MAXPR 189
MAXSSAUD 189
MAXSSWTO 189
security 216
secondary indexes 58
truly unique 58
security 216, 230
and SCU execution 216
for audit trail table 230
SEG keyword 30
SEGM macro 106
Segment exit routine 28
internal segments and 28
segments 43, 107, 129
DEDB direct dependent 43
specifying EXIT with logical child 107
specifying those to be propagated 129
SEQ sub-parameter 59
SET CURRENT PACKAGESET statement 88
SETS call 86
setup phase 3, 234
administrator tasks 3
performance 234
share control 99, 193
Sign-On Authorization exit 148
single-byte binary field 77
single-precision floating point number 78
small integer field 77
SMF (System Management facilities) 228
SMFPRMxx member 228
SQL 88
SQL *(continued)*
SQL communications area 88
SQL (structured query language) 101, 117, 179, 187, 188, 224
error codes 179, 188
errors 187
protecting tables from updates 117
remote updates to propagated tables 101
repair statements 224
updates in non-IMS environment 101
SQL (Structured Query Language) 22, 88, 208
COMMIT statement 88
propagation of calls 22
resident update modules 208
ROLLBACK statement 88
SET CURRENT PACKAGESET statement 88
SQLCA 88, 180
SQLERRMC field 180
SQLSTATEs 179, 188
SSM member 246, 248
state of synchronous propagation 192
active 192
statements 59, 88, 100, 113, 119, 123, 124, 136, 153, 155, 156, 157, 160, 165, 169, 175, 176, 178, 188, 189, 192, 193, 196, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 215, 216, 217, 223, 227, 235, 236, 238, 243, 244, 247
//DDOTV02 247
//EKYIN 244
//EKYLOG 244
//EKYPRINT 244
//EKYRESLB 243
//EKYSNAP 244
//EKYSTATF 243
//EKYTRACE 244
//IEFRDER 178
/DBDUMP 196
/DBRECOVERY 196
ACTIVATE 160, 192, 196
ALIAS 156
ALLOWPROPOFF 193, 203, 215, 216
ALTER TABLE 119, 153, 176
CHECK 238
COMMIT 88
COPY 113
CREATE ALIAS 155, 157
CREATE SYNONYM 155, 157
CREATE TABLE 119, 153, 176
DEACTIVATE 192, 196, 198, 199
DELETE 136
DENYPROPOFF 203, 215
DISPLAY 203
EDEACTIVATE 192, 196
ERRCTL 189, 204
ERROPT 204
ESTOP 165, 202
EXCLUDE 169
EXTRACT 124
FIELD 59, 123
INIT STATF 205
INIT VLF 236
PROP LOAD 100, 206, 217
PROP OFF 175, 206, 216
statements *(continued)*
PROP SUSP 176, 188, 192, 206
READOFF 200, 201
READOFF DB2DB 215
READON 200, 201
READON DB2DB 215
repair 223
RESET 202
RESIDENT 209
RESIDENT SQLU 209, 235
ROLLBACK 88
RUP and HUP control 205
RUP/HUP control 209
SET CURRENT PACKAGESET 88
SUSPEND 196
SYNONYM 156
TRACE 207, 227
TRACEOFF 205
TRACEON 205
TRDEST 208
states of synchronous propagation 191, 192
emergency stopped 191
inactive 192
not emergency stopped 191
suspended 192
Status Change utility 160
See SCU 160
status codes, IMS 179, 188
status file 92, 243
storage requirements 253, 255
synchronous propagation 253, 255
SUBMIT command 124, 131, 162, 163
suboptions of CASCADE 108
subsequent bind 154
suppressing messages 189
SUSPEND control statement 53, 143, 196
orderly status changes 196
privileges required 143
use with unqualified table names 53
suspended state 192
synchronization 85
point coordinator 85
synchronous propagation 3, 85, 88, 102, 109, 115, 118, 175, 190, 191, 209, 211, 218, 221, 233, 234, 253, 255
and database maintenance 211, 218
and the CCU 221
application programs, considerations 85, 88
binding DB2 plans, re-bind 118
controlling synchronous propagation states 191
coordinating availability between IMS and DB2 102
examples of EXIT keyword 109
initial bind 115
performance 233, 234
processing during 175, 190
storage requirements 253, 255
synchronous propagation states and controlling synchronous propagation 209
tasks 3
synchronous propagation states 191, 192, 209

synchronous propagation states *(continued)*
See also states of synchronous propagation
description 191, 209
SYNONYM statement 156
SYS1.MANx data sets 229
SYS1.PARMLIB 92, 228
and recording SMF records 228
and VLF 92
audit trail table 228
Sysplex 93, 94
JCL changes 94
use of GMTS 93
system information 95
System Management facilities 228
See SMF 228

T

table qualification 117
tables 89, 92, 95, 116, 128, 129, 131, 140, 228, 231
audit 95
audit trail 140, 228
creating DB2, synchronous 116
DPRIFLD 128
DPRIPR 128, 131
DPRISEG 128
DPRITAB table 128
DPRIWHR 128
IMS DPROP directory 89, 92, 140, 231
MVG input 95, 128, 131, 140
defining propagation requests using 128, 131
description 95
privileges for 140
specifying those to be propagated 129
TABQUAL2 parameter 133
tasks 3
synchronous propagation 3
test environment 95
times, mapping and conversion between 83
timestamps 83, 214
mapping and conversion between 83
recovery 214
tools 227, 231, 238
diagnostic 227, 231
for monitoring 238
IMSASAP II 238
IMSPARS 238
trace class 6 119
TRACE control statement 207, 227
TRACEOFF control statement 205
TRACEON control statement 205
tracing 205, 207, 208, 227
controlling 207, 208
trace facilities 227
turning on an off 205
trademarks 261
TRANSACT macro 113, 183
transient storage requirements 254
TRANSLATE request 88
TRDEST control statement 208
truly unique secondary indexes 58
TW (two-way) propagation 11, 12
See two-way synchronous propagation
two-way propagation 58
unique indexes and 58
two-way synchronous propagation 57, 117, 145, 236
defining RIRs for 57
granting privileges when doing 145
implementing non-matching RIRs for 57
propagation phase 236
protecting tables from updates 117

U

U3303 abend 182
UCF (Utility Control Facility) 215
UIM (user input manager) 121
unavailable resources 186, 187
unidirectional relationships 45
unique indexes 57, 58
unique key field 59
UNLIMITED keyword 204
unqualified table names 51
user asynchronous propagation 237
performance 237
user mapping 176, 177
and HUP processing 177
and RUP processing 176
user mapping cases 30, 127
and DataRefresher 127
description 30
user mapping PR 12, 17
See PRTYPE=U 12
utilities 52, 100, 101, 129, 138, 142, 160, 165, 166, 172, 178, 185, 194, 205, 208, 212, 213, 214, 215, 217, 219, 225, 228, 234, 235, 238
AUDU (Audit Extract utility) 142, 228
binding plans of 142
problem determination 228
Batch Backout 178, 185, 212
binding plans of 142
CCU 142, 219, 225
binding plans of 142
description 219, 225
database updates with 100
DB2 Load 217
DB2 Repair 215
DFSBBO00 212
DFSERA10 208, 228
DFSURGU0 166, 214
DLU 165, 172
EXPLAIN 238
Fast Reload 234
File Select and Formatting Print 208, 228
HD Reload 234
HD Reorganization Unload 214
HSSR DB Unload 166
Image Copy 214
IMS Monitor 238
Load 234
MVGU 129, 138, 142
binding plans of 142
utilities *(continued)*
MVGU *(continued)*
control statements 138
executing 129
Performance Monitor 238
Prefix Resolution and Update 234
Quiesce 213
running IMS DPROP 142
RUNSTATS 235
SCU (Status Change utility) 52, 142, 160, 194, 205
binding plans of 142
description 194, 205
using 160
using with unqualified table names 52
table updates with 101
UCF 215

V

VALIDATE parameter 151, 156
VALIDATE(BIND) parameter 236
Validation exit routine 117
VALIDPROC clause 117
variable-length character field 78
variable-length graphic field 78
variable-length segments 45
version ID 111
VERSION keyword 111
VIRTUAL delete rule 55, 258
Virtual Lookaside facility 92
See VLF 92
virtual storage requirements 253
virtually paired segment types 44, 172
VLF (virtual lookaside facility) 189
and error messages 189
VLF (Virtual Lookaside facility) 92, 93
initializing objects 93
of 92
refreshing or recreating objects 93
requirements 92
VLF (Virtual Lookaside Facility) 205, 233
benefits of 233
initializing objects 205
VSAM Zapper program 215

W

wait-for-input (WFI) transactions 236
WFI (wait-for-input) transactions 236
WHERE clause 37, 40, 41
description 37, 41
operators 40

X

XRF (Extended Recovery facility) 100

Z

ZAP function 215
Zapper program of VSAM 215

Readers' Comments — We'd Like to Hear from You

IBM IMS DataPropagator for z/OS
Administrators Guide for Synchronous Propagation
Version 3 Release 1

Publication No. SC27-1215-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

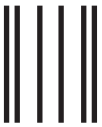
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC27-1215-01

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
H150/090
555 Bailey Avenue
San Jose, CA
USA 95141-9989

Fold and Tape | **Please do not staple** | Fold and Tape

SC27-1215-01

Cut or Fold
Along Line

File Number: 5655-E52

Printed in USA

SC27-1215-01