IBM DB2 Information Integrator

SQL Replication Guide and Reference

Version 8.2

SC27-1121-02

IBM DB2 Information Integrator

SQL Replication Guide and Reference

Version 8.2

SC27-1121-02

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 541.

You can order IBM publications online or through your local IBM representative:

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

About this book ix
Who should read this book ix
How to use this book ix
Conventions and terminology used in this book . . xi
How to read syntax diagrams xii
Road map xiii

What's new for DB2 replication for Version 8? xv
What's new in Version 8.2? xv
New replication solutions xv
New function xv
What's new in Version 8.1.4? xvi
New function xvi
Performance improvements xvi
What's new in Version 8 fix pack 2? xvi
Usability improvements xvi
Performance improvements xvi
New function xvi
Changes to control tables xvii
What's new in Version 8.1?. xvii
Usability improvements xvii
Performance improvements xviii
New user interface. xix
New function xix
Serviceability improvements xxii
Changes to replication system commands . . . xxii
Changes to control tables xxiii
Functions no longer supported xxv

Part 1. Replication guide 1

Chapter 1. Planning for SQL replication 3
Migration planning 3
Memory planning 3
Memory used by the Capture program 3
Memory used by the Apply program 5
Memory used by the Replication Alert Monitor . . 5
Storage planning 5
Planning log impact 6
Planning the storage requirements of target tables and control tables. 7
Planning storage requirements for temporary files 8
Planning for conflict detection 10
Planning for non-DB2 relational sources 10
Planning transaction throughput rates for Capture triggers 10
Planning the log impact for non-DB2 relational source servers 11
Planning locks for Oracle source servers 11
Planning coexistence of pre-existing triggers with Capture triggers 11
Planning for code page translation. 11
Replicating data between databases with compatible code pages. 12
Configuring national language support (NLS) for replication 12
Replication planning for DB2 UDB for z/OS . . . 13
Performance tuning. 13

Chapter 2. Configuring servers for SQL replication 15
Controlling access to replication servers 15
Connectivity requirements for replication . . . 15
Authorizing user IDs for replication 17
Authorization requirements for administration . 17
Authorization requirements for the Capture program 18
Authorization requirements for Capture triggers on non-DB2 relational databases 19
Authorization requirements for the Apply program 19
Authorization requirements for the Replication Alert Monitor. 21
Storing user IDs and passwords for replication (Linux, UNIX, Windows) 22
Setting up the replication control tables 22
Creating control tables (Linux, UNIX, Windows) 22
Creating control tables (z/OS) 22
Creating control tables (OS/400) 23
Creating control tables for non-DB2 relational sources 23
Creating multiple sets of Capture control tables 24
Capture control tables on multiple database partitions 24
Setting up the replication programs 25
Setting up the replication programs (Linux, UNIX, Windows) 25
Setting up the Capture and Apply programs (OS/400) 28
Setting up the replication programs (z/OS). . . 30
Capture for multiple database partitions. . . . 30
Setting up journals (OS/400) 30
Creating journals for source tables (OS/400) . . 30
Managing journals and journal receivers (OS/400) 32

Chapter 3. Registering tables and views as SQL replication sources . . . 35
Registering DB2 tables as sources 35
Registering non-DB2 relational tables as sources . . 37
Registration options for source tables. 38
Registering a subset of columns (vertical subsetting). 39
Full-refresh copying and change-capture replication 39
After-image columns and before-image columns 41
Before-image prefix. 43

Stop the Capture program on error 43
How the Capture program stores updates . . . 44
Preventing the recapture of changes (update-anywhere replication) 45
Setting conflict detection (update-anywhere replication) 49
Registering tables that use remote journaling (OS/400) 51
Using relative record numbers (RRN) instead of primary keys (OS/400) 51
How views behave as replication sources 52
Views over a single table 52
Views over a join of two or more tables 52
Registering views of tables as sources 54
Maintaining CCD tables as sources (IMS) 55

Chapter 4. Subscribing to sources for SQL replication 57
Planning how to group sources and targets 57
Planning the number of subscription-set members 58
Planning the number of subscription sets per Apply qualifier 58
Creating subscription sets 59
Processing options for subscription sets 61
Specifying whether the set is active 61
Specifying how many minutes worth of data the Apply program retrieves 61
Deciding how the Apply program loads target tables that have referential integrity 63
Specifying how the Apply program replicates changes for members in the set 64
Defining SQL statements or stored procedures for the subscription set 65
Scheduling the replication of a subscription set 65
Mapping source tables and views to target tables and views within a subscription set 67
Selecting a target type 69
Defining read-only target tables 71
Defining middle tiers in a multi-tier configuration 75
Defining read-write targets (update-anywhere) . 77
Using an existing table as the target table . . . 79
Common properties for all target table types . . . 80
Source columns that you want applied to the target 80
Source rows that you want applied to the target 80
How source columns map to target columns . . 81
Target key 82
How the Apply program updates the target key columns with the target-key change option . . . 83

Chapter 5. Replicating special data types in SQL replication 85
General data restrictions for replication 85
Replicating large objects 85
Replicating DATALINK values 86
Setting up and using the ASNDLCOPY exit routine 88
Setting up and using DLFM_ASNCOPYD (Linux, UNIX, Windows) 90
Setting up and using ASNDLCOPYD (OS/400) 91

Chapter 6. Subsetting data in an SQL replication environment 93
Subsetting data during registration 93
Subsetting source data using views 94
Defining triggers on CD tables to prevent specific rows from being captured (Linux, UNIX, Windows, z/OS). 94
Subsetting data during subscription 95

Chapter 7. Manipulating data in an SQL replication environment 97
Enhancing data using stored procedures or SQL statements 98
Mapping source and target columns that have different names 98
Creating computed columns 99

Chapter 8. Customizing and running replication SQL scripts for SQL replication 101

Chapter 9. Operating the Capture program for SQL replication 103
Default operational parameters for the Capture program 103
Changing operational parameters for the Capture program 105
Starting the Capture program (Linux, UNIX, Windows, z/OS) 107
add_partition (Linux, UNIX, Windows). . . . 108
autoprune (Linux, UNIX, Windows, z/OS) . . 108
autostop (Linux, UNIX, Windows, z/OS) . . . 109
capture_path (Linux, UNIX, Windows, z/OS) 109
capture_schema (Linux, UNIX, Windows, z/OS) 110
capture_server (Linux, UNIX, Windows, z/OS) 110
commit_interval (Linux, UNIX, Windows, z/OS) 110
lag_limit (Linux, UNIX, Windows, z/OS) . . . 111
logreuse (Linux, UNIX, Windows, z/OS) . . . 111
logstdout (Linux, UNIX, Windows, z/OS) . . . 111
memory_limit (Linux, UNIX, Windows, z/OS) 112
monitor_interval (Linux, UNIX, Windows, z/OS) 112
monitor_limit (Linux, UNIX, Windows, z/OS) 112
prune_interval (Linux, UNIX, Windows, z/OS) 112
retention_limit (Linux, UNIX, Windows, z/OS) 113
sleep_interval (Linux, UNIX, Windows, z/OS) 114
startmode (Linux, UNIX, Windows, z/OS). . . 114
term (Linux, UNIX, Windows, z/OS) 115
trace_limit (Linux, UNIX, Windows, z/OS) . . 115
Starting the Capture program (OS/400). 115
Altering the behavior of a running Capture program 116
Changing the operating parameters in the Capture parameters table 117
Stopping the Capture program 118

Suspending Capture (Linux, UNIX, Windows, z/OS) 118
Resuming Capture (Linux, UNIX, Windows, z/OS) 119
Reinitializing Capture 119

Chapter 10. Operating the Apply program for SQL replication 121
Default operational parameters for the Apply program 121
Changing operational parameters for the Apply program 122
Starting the Apply program (Linux, UNIX, Windows, z/OS) 123
apply_path (Linux, UNIX, Windows, z/OS) . . 124
apply_qual (Linux, UNIX, Windows, z/OS) . . 125
control_server (Linux, UNIX, Windows, z/OS) 125
copyonce (Linux, UNIX, Windows, z/OS) . . . 125
db2_subsystem (z/OS) 126
delay (Linux, UNIX, Windows, z/OS) 126
errwait (Linux, UNIX, Windows, z/OS) . . . 126
inamsg (Linux, UNIX, Windows, z/OS) . . . 127
loadxit (Linux, UNIX, Windows, z/OS). . . . 127
logreuse (Linux, UNIX, Windows, z/OS) . . . 127
logstdout (Linux, UNIX, Windows, z/OS) . . . 127
notify (Linux, UNIX, Windows, z/OS) 128
opt4one (Linux, UNIX, Windows, z/OS) . . . 128
pwdfile (Linux, UNIX, Windows). 128
sleep (Linux, UNIX, Windows, z/OS) 128
spillfile (Linux, UNIX, Windows, z/OS) . . . 129
sqlerrcontinue (Linux, UNIX, Windows, z/OS) 129
term (Linux, UNIX, Windows, z/OS) 130
trlreuse (Linux, UNIX, Windows, z/OS) . . . 130
Starting the Apply program (OS/400) 131
Changing the operating parameters in the Apply parameters table (Linux, UNIX, Windows, z/OS) . 132
Stopping the Apply program 133
Modifying the ASNDONE exit routine (Linux, UNIX, Windows, z/OS) 133
Modifying the ASNDONE exit routine (OS/400) 134
Refreshing target tables using the ASNLOAD exit routine 135
Refreshing target tables with the ASNLOAD exit routine (Linux, UNIX, Windows) 136
Refreshing target tables with the ASNLOAD exit routine (z/OS) 137
Customizing ASNLOAD exit behavior (Linux, UNIX, Windows, z/OS) 138
Refreshing target tables with the ASNLOAD exit routine (OS/400) 140

Chapter 11. Monitoring replication with the Replication Alert Monitor . . 143
Monitoring replication with the Replication Alert Monitor—Overview 143
The Replication Alert Monitor 143
Alert conditions and notifications for the Replication Alert Monitor 145
Alert conditions and notifications for the Replication Alert Monitor—Overview 145
Alert conditions for the Replication Alert Monitor 146
E-mail notifications for replication alert conditions 149
The ASNMAIL exit routine for sending alerts in replication 150
Setting up the Replication Alert Monitor 151
Setting up the Replication Alert Monitor . . . 151
Creating control tables for the Replication Alert Monitor 151
Defining contact information for the Replication Alert Monitor 152
Operating the Replication Alert Monitor 153
Operating the Replication Alert Monitor . . . 153
Creating monitors for replication or publishing 154
Selecting alert conditions for the Replication Alert Monitor 155
Starting monitors 156
Reinitializing monitors 156
Default values of the parameters that are used to operate the Replication Alert Monitor . . . 157
Descriptions of the parameters that are used to operate the Replication Alert Monitor 158
Setting parameters for the Replication Alert Monitor 160
Stopping a monitor 163

Chapter 12. On-demand reporting for SQL replication 165
Checking for current status of replication programs (Linux, UNIX, Windows, z/OS) 165
Checking the status of the Capture and Apply journal jobs (OS/400). 166
Reviewing historical data for trends 167
Reviewing Capture program messages 168
Examining Capture program throughput . . . 168
Displaying latency of data processed by the Capture program 169
Reviewing Apply program messages 169
Examining Apply program throughput 170
Displaying the average length of time taken to replicate transactions 170
Reviewing Monitor program messages 171
Monitoring the progress of the Capture program (OS/400) 171

Chapter 13. Making changes to an SQL replication environment. 173
Registering new objects 173
Changing registration attributes for registered objects. 174
Adding columns to source tables 174
Stop capturing changes for registered objects . . . 177
Reactivating registrations 178
Removing registrations 179
Changing Capture schemas. 179
Creating new subscription sets 182
Adding new subscription-set members to existing subscription sets 182

Disabling subscription-set members from existing subscription sets 183
Enabling subscription-set members to existing subscription sets 183
Changing attributes of subscription sets 183
Changing subscription set names 184
Splitting a subscription set 186
Merging subscription sets 189
Changing Apply qualifiers of subscription sets . . 192
Deactivating subscription sets 194
Removing subscription sets. 195
Coordinating replication events with database application events 196
Setting an event END_SYNCHPOINT using the USER type signal 196
Creating journal signal tables for remote journaling 197
Using the Capture CMD STOP signal 198
Performing a CAPSTART handshake signal outside of the Apply program 201
Performing a CAPSTOP signal 202
Promoting your replication configuration to another system 203

Chapter 14. Maintaining an SQL replication environment 205
Maintaining your source systems 205
Maintaining source objects 205
Maintaining and retaining source logs and journal receivers 205
Maintaining your control tables 209
Using the RUNSTATS utility (Linux, UNIX, Windows, z/OS) 210
Rebinding packages and plans (Linux, UNIX, Windows, z/OS) 210
Reorganizing your control tables 210
Pruning your control tables. 212
Preventing replication failures and recovering from errors 215
Maintaining your target tables. 216

Part 2. Replication Center 217

Chapter 15. Using the Replication Center for SQL replication 219
Prerequisites for the Replication Center. 220
Configuring the Replication Center for host RDBMSs 221
Starting the Replication Center 221
Using the Replication Center launchpad for SQL replication 222
Managing user IDs and passwords for the Replication Center. 223
Creating replication profiles 224
Creating control-table profiles 224
Creating source-object profiles 225
Creating target-object profiles 226
Creating replication control tables 227
Creating Capture control tables 227
Creating Apply control tables 228
Creating Monitor control tables 228
Adding servers to the Replication Center 229
Enabling a database for change capture (UNIX and Windows) 230
Registering sources 231
Creating subscription sets 232
Defining the information for the subscription set 233
Mapping sources to targets 233
Scheduling the subscription set 235
Adding SQL statements or stored procedures to the subscription set 235
Activating or deactivating subscription sets . . . 235
Promoting replication objects 236
Promoting registered tables or views 236
Promoting subscription sets 237
Forcing a full refresh of target tables 237
Removing or deleting replication definitions . . . 238
Operating the Capture program 238
Operating the Apply program 239
Operating the Replication Alert Monitor 239

Chapter 16. Basic SQL replication scenario: DB2 for Windows 243
Before you begin 243
Planning this scenario 244
Replication source 244
Replication target 244
Replication options 245
Setting up the replication environment for this scenario 246
Step 1: Create replication control tables for the Capture program 246
Step 2: Enable the source database for replication 246
Step 3: Register a replication source 247
Step 4: Create replication control tables for the Apply program. 249
Step 5: Create a subscription set and a subscription-set member. 250
Step 6: Create an Apply password file 254
Step 7: Replicate the scenario data 255
Operating in a replication environment. 256
Step 1: Update the source table 257
Step 2: View status for the Capture program . . 257
Step 3: View status for the Apply program . . 258
Step 4: Stop the Capture and Apply programs 259
Monitoring replication 260
Step 1: Create replication control tables for the Monitor program 260
Step 2: Create a contact for replication alerts . . 262
Step 3: Select alert conditions for the Capture program 262
Step 4: Select alert conditions for the Apply program 263
Step 5: Start the Replication Alert Monitor for a monitor qualifier 264

Part 3. Replication reference . . . 267

Chapter 17. Naming rules for SQL replication objects 269

Chapter 18. System commands for SQL replication (Linux, UNIX, Windows, z/OS) 271
asnacmd: Operating Apply 271
asnanalyze: Operating the Analyzer 273
asnapply: Starting Apply 276
asncap: Starting Capture. 282
asnccmd: Operating Capture 288
asnmcmd: Working with a running Replication Alert Monitor 292
asnmon: Starting a Replication Alert Monitor. . . 295
asnpwd: Creating and maintaining password files 299
asnscrt: Creating a DB2 replication service to start the replication programs. 302
asnsdrop: Dropping DB2 replication services . . . 305
asnslist: Listing DB2 replication services 306
asntdiff: Comparing data in source and target tables 307
asntrc: Operating the replication trace facility. . . 309
asntrep: Repairing differences between source and target tables 316

Chapter 19. System commands for SQL replication (OS/400) 319
ADDDPRREG: Adding a DPR registration (OS/400) 319
ADDDPRSUB: Adding a DPR subscription set (OS/400) 327
ADDDPRSUBM: Adding a DPR subscription-set member (OS/400) 343
ANZDPR: Operating the Analyzer (OS/400) . . . 352
CHGDPRCAPA: Changing DPR Capture attributes (OS/400) 355
CRTDPRTBL: Creating the replication control tables (OS/400) 360
ENDDPRAPY: Stopping Apply (OS/400) 361
ENDDPRCAP: Stopping Capture (OS/400) . . . 364
GRTDPRAUT: Authorizing users (OS/400) . . . 366
INZDPRCAP: Reinitializing DPR Capture (OS/400) 374
OVRDPRCAPA: Overriding DPR capture attributes (OS/400) 375
RMVDPRREG: Removing a DPR registration (OS/400) 380
RMVDPRSUB: Removing a DPR subscription set (OS/400) 381
RMVDPRSUBM: Removing a DPR subscription-set member (OS/400) 383
RVKDPRAUT: Revoking authority (OS/400) . . . 384
STRDPRAPY: Starting Apply (OS/400) 386
STRDPRCAP: Starting Capture (OS/400) 393
WRKDPRTRC: Using the DPR trace facility (OS/400) 400

Chapter 20. Operating the SQL replication programs (z/OS) 405
Using JCL or system-started tasks to operate the replication programs (z/OS) 405
Using JCL to operate replication programs . . 405
Using system-started tasks to operate replication programs. 407
Using MVS Automatic Restart Manager (ARM) to automatically restart replication programs (z/OS) . 407
Migrating your replication environment to data-sharing mode (z/OS) 408

Chapter 21. Using the Windows Service Control Manager to issue system commands for SQL replication (Windows). 409
Creating a replication service 409
Operating a replication service 410
Dropping a replication service. 410

Chapter 22. Scheduling SQL replication programs on various operating systems 413
Scheduling programs on UNIX and Linux operating systems 413
Scheduling programs on Windows operating systems 413
Scheduling programs on z/OS operating systems 413
Scheduling programs on the OS/400 operating system 414

Chapter 23. How the SQL replication components communicate. 415
The Replication Center, the Capture program or triggers, and the Apply program 415
The Capture program and the Apply program . . 416
The Capture triggers and the Apply program. . . 417
The Replication Center and the Replication Alert Monitor 418
The Replication Alert Monitor, the Capture program, and the Apply program 419

Chapter 24. Table structures for SQL replication. 421
Tables at a glance 421
List of tables used at the Capture control server 428
List of tables used at the Apply control server . . 431
List of control tables at the Monitor control server 432
List of tables used at the target server 433
Tables at the Capture control server and their column descriptions 433
ASN.IBMSNAP_CAPSCHEMAS 434
schema.IBMSNAP_AUTHTKN (OS/400) . . . 434
schema.IBMSNAP_CAPENQ (UNIX, Windows, z/OS) 435
schema.IBMSNAP_CAPMON 436
schema.IBMSNAP_CAPPARMS 437
schema.IBMSNAP_CAPTRACE (DB2 only). . . 440
schema.CCD_table (non-DB2) 441
schema.CD_table. 442
schema.IBMSNAP_PARTITIONINFO 443

schema.IBMSNAP_PRUNCNTL 444
schema.IBMSNAP_PRUNE_LOCK 446
schema.IBMSNAP_PRUNE_SET 447
schema.IBMSNAP_REG_EXT (OS/400) 447
schema.IBMSNAP_REGISTER 449
schema.IBMSNAP_REG_SYNCH (non-DB2 relational) 455
schema.IBMSNAP_RESTART 456
schema.IBMSNAP_SEQTABLE (Informix) . . . 458
schema.IBMSNAP_SIGNAL 458
schema.IBMSNAP_UOW 461
Tables at the Apply control server and their column descriptions 463
ASN.IBMSNAP_APPENQ 463
ASN.IBMSNAP_APPLY_JOB (OS/400) 464
ASN.IBMSNAP_APPPARMS 465
ASN.IBMSNAP_APPLYTRACE 467
ASN.IBMSNAP_APPLYTRAIL. 468
ASN.IBMSNAP_SUBS_COLS 473
ASN.IBMSNAP_SUBS_EVENT 475
ASN.IBMSNAP_SUBS_MEMBR 475
ASN.IBMSNAP_SUBS_SET 480
ASN.IBMSNAP_SUBS_STMTS. 485
Tables at the Monitor control server and their column descriptions 487
IBMSNAP_ALERTS table 487
IBMSNAP_CONDITIONS table 488
IBMSNAP_CONTACTGRP table 493
IBMSNAP_CONTACTS table 494
IBMSNAP_MONENQ table. 495
IBMSNAP_GROUPS table 495
IBMSNAP_MONPARMS table. 495
IBMSNAP_MONSERVERS table 497
IBMSNAP_MONTRACE table. 498
IBMSNAP_MONTRAIL table 499
Tables at the target server and their column descriptions 501
Base aggregate table 501
Change aggregate table 501
Consistent-change data (CCD) table 502
Point-in-time table. 504
Replica table 505
User copy table. 505

Appendix A. UNICODE and ASCII encoding schemes for SQL replication (z/OS). 507
Choosing an encoding scheme. 507
Setting Encoding Schemes 507

Appendix B. How the Capture program processes journal entry types for SQL replication (iSeries) . . 509

Appendix C. Starting the SQL replication programs from within an application (Linux, UNIX, Windows) . . 511

Glossary 513
Glossary 513

Index 525

Accessibility. 539
Keyboard input and navigation 539
Keyboard input. 539
Keyboard navigation 539
Keyboard focus. 539
Accessible display 539
Font settings. 539
Non-dependence on color 540
Compatibility with assistive technologies 540
Accessible documentation 540

Notices 541
Trademarks 543

Contacting IBM 545
Product information 545
Comments on the documentation. 545

About this book

This book describes how to plan, set up, maintain, and monitor a data replication environment using DB2 replication, which is the focus of the book. This book contains the guidance and reference information for the SQL replication component that is introduced in *IBM DB2 Information Integrator Introduction to Replication and Event Publishing*.

SQL replication, also referred to as DB2 replication, is a type of replication that uses SQL to replicate data between systems. The term is used to differentiate this type of replication from Q replication, which replicates data through message queues.

Who should read this book

This book is written for database administrators, LAN administrators, and others who must set up and maintain a data replication environment in an SQL environment. You should be familiar with standard SQL database terminology, have a working knowledge of the operating systems that will be involved in replication, and have experience with database design, database administration, database security, server connectivity, and networking. You should understand the applications in your environment and how they manipulate the data that you want to replicate using SQL queries and commands. You should be familiar with basic replication concepts and components.

How to use this book

Most sections in this book pertain to SQL replication function for all operating-system environments. There are some sections that contain operating-system-specific information.

The organization and content of this book have changed since the last release. This book contains three parts:

- Part 1, "Replication guide," on page 1 describes how to plan, set up, run, and maintain your replication environment. It includes the following chapters:
  - Chapter 1, "Planning for SQL replication," on page 3 describes how to plan and design your replication environment.
  - Chapter 2, "Configuring servers for SQL replication," on page 15 describes how to prepare your environment for replication.
  - Chapter 3, "Registering tables and views as SQL replication sources," on page 35 describes what you need to know to register replication sources.
  - Chapter 4, "Subscribing to sources for SQL replication," on page 57 describes what you need to know to create subscription sets and add members to subscription sets.
  - Chapter 5, "Replicating special data types in SQL replication," on page 85 describes the replication options for LOB and DATALINK values in source tables.
  - Chapter 6, "Subsetting data in an SQL replication environment," on page 93 describes how to customize what data is captured and applied to the target as well as how the data is applied to the target.

- Chapter 7, “Manipulating data in an SQL replication environment,” on page 97 describes how to use the Capture program or the Apply program to manipulate source data.
- Chapter 8, “Customizing and running replication SQL scripts for SQL replication,” on page 101 describes how to run SQL in your replication environment.
- Chapter 9, “Operating the Capture program for SQL replication,” on page 103 describes how to operate the Capture program for all operating-system environments.
- Chapter 10, “Operating the Apply program for SQL replication,” on page 121 describes how to operate the Apply program for all operating-system environments.
- Chapter 11, “Monitoring replication with the Replication Alert Monitor,” on page 143 describes how to use the Replication Alert Monitor to monitor your replication environment.
- Chapter 12, “On-demand reporting for SQL replication,” on page 165 describes how to generate and view reports about your replication environment on demand.
- Chapter 13, “Making changes to an SQL replication environment,” on page 173 describes how to make day-to-day changes in your replication environment.
- Chapter 14, “Maintaining an SQL replication environment,” on page 205 explains how to maintain your source systems, control tables, and target tables.

• Part 2, “Replication Center,” on page 217 describes the graphical user interface for replication. It includes the following chapters:
  - Chapter 15, “Using the Replication Center for SQL replication,” on page 219 describes the Replication Center.
  - Chapter 16, “Basic SQL replication scenario: DB2 for Windows,” on page 243 describes how to use the Replication Center to perform a simple replication scenario using sample data.
• Part 3, “Replication reference,” on page 267 describes replication commands and replication table structures. It includes the following chapters:
  - Chapter 17, “Naming rules for SQL replication objects,” on page 269 describes how to specify valid names for replication objects.
  - Chapter 18, “System commands for SQL replication (Linux, UNIX, Windows, z/OS),” on page 271 describes commands that experienced DB2 replication users can use instead of the Replication Center for operating replication on the Linux, UNIX, Windows, and z/OS operating systems.
  - Chapter 19, “System commands for SQL replication (OS/400),” on page 319 describes the commands that you can use if you want to set up, administer, and maintain replication locally on the OS/400 operating system.
  - Chapter 20, “Operating the SQL replication programs (z/OS),” on page 405 describes how to start and operate the replication programs using JCL or system-started tasks on the z/OS operating system.
  - Chapter 21, “Using the Windows Service Control Manager to issue system commands for SQL replication (Windows),” on page 409 describes how to create services to operate the Replication programs on the Windows operating system.
  - Chapter 22, “Scheduling SQL replication programs on various operating systems,” on page 413 describes how to schedule the Capture, Apply, and Replication Alert Monitor programs on various operating systems.

  - Chapter 23, "How the SQL replication components communicate," on page 415 describes how the replication components use the control tables to communicate with each other.
  - Chapter 24, "Table structures for SQL replication," on page 421 describes the table structures for the SQL replication tables that reside on the various SQL replication servers.
- Appendixes contain supplemental information that you might find useful.

Conventions and terminology used in this book

This book uses these highlighting conventions:

- **Boldface type** indicates commands or user interface controls such as names of fields, folders, icons, or menu choices.
- `Monospace type` indicates examples of text that you enter exactly as shown.
- *Italic type* indicates variables that you should replace with a value. It is also used to indicate book titles and for emphasis of words.

This book uses standard terminology for database, connectivity, copying, SQL, and LAN concepts. All the replication concepts used in this book are defined in the glossary.

Unless otherwise specified, the following meanings are assumed:

UNIX UNIX refers to DB2 Universal Database for all UNIX operating systems (such as UNIX, HP UX, and AIX).

Linux Linux refers to DB2 Universal Database for Linux.

Windows
Windows refers to DB2 Universal Database for Windows.

OS/400
OS/400 refers to DB2 DataPropagator for iSeries.

z/OS z/OS refers to DB2 Universal Database for z/OS and OS/390. z/OS is the next generation of the OS/390 operating system, and it also includes UNIX System Services (USS) on z/OS.

iSeries
iSeries refers to both AS/400 and iSeries servers. iSeries is the next generation of AS/400 servers. The OS/400 operating system runs on both AS/400 and iSeries servers.

SQL replication
SQL replication, also referred to as DB2 replication, is one of two types of data replication developed for DB2. It is used to differentiate replication through SQL from Q replication, which is replication through message queues. The Capture program reads the DB2 recovery log for changes to a source table that you specify. The program saves transactions in staging tables that are read and applied to targets by the Apply program in parallel.

Q replication
Q replication is a high-volume, low-latency replication solution that uses WebSphere MQ message queues to transmit transactions between source and target databases or subsystems. The Q Capture program reads the DB2 recovery log for changes to a source table that you specify. The program then sends transactions as messages over queues, where they are read and applied to targets by the Q Apply program in parallel.

event publishing
In event publishing, changes to source tables are translated into XML messages and sent over WebSphere MQ queues to a user application of your choice. Event publishing uses only the Q Capture program, not the Q Apply program.

For example, the section entitled *Starting the Apply program (Linux, UNIX, Windows, z/OS)* explains how to start the Apply program from DB2 Universal Database for Linux and for all UNIX operating systems, DB2 Universal Database for Windows, or DB2 Universal Database for z/OS and OS/390. Also, the section entitled *Starting the Apply program (OS/400)* explains how to start the Apply program if you are using DB2 DataPropagator for iSeries.

How to read syntax diagrams

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

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

  The ▶▶— symbol indicates the beginning of a statement.

  The —▶ symbol indicates that the statement syntax is continued on the next line.

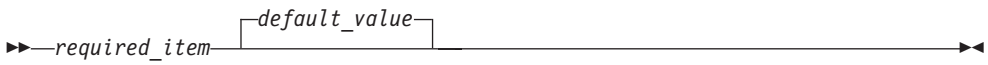
  The ▶— symbol indicates that a statement is continued from the previous line.

  The —▶◀ symbol indicates the end of a statement.

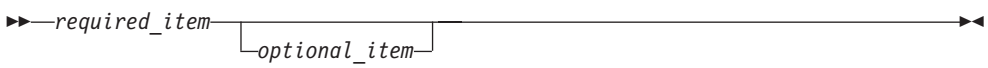
  Diagrams of syntactical units other than complete statements start with the ▶— symbol and end with the —▶ symbol.
- Keywords, their allowable synonyms, and reserved parameters, are either shown in uppercase or lowercase, depending on the operating system. These items must be entered exactly as shown. Variables appear in lowercase italics (for example, *column-name*). They represent user-defined parameters or suboptions.

  When entering commands, separate the parameters and keywords by at least one space if there is no intervening punctuation.
- Enter punctuation marks (slashes, commas, periods, parentheses, quotation marks, equal signs, and so on) and numbers exactly as given.
- Footnotes are shown by a number in parentheses, for example, (1).
- Required items appear on the horizontal line (the main path).

- A parameter's default value is displayed above the path:

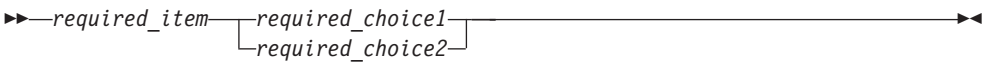


- Optional items appear below the main path.

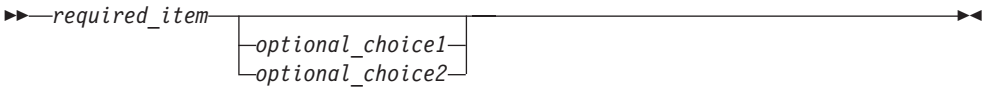


- If you can choose from two or more items, they appear vertically, in a stack.

  If you *must* choose one of the items, one item of the stack appears on the main path.



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



Road map

This section identifies other sources of information about DB2 replication that you might find useful.

Table 1. Information road map.

If you want to ...	Refer to ...
Access information about DataPropagator	DB2 DataPropagator at www.ibm.com/software/data/dpropr/
Learn about Q replication, SQL replication, and event publishing	*IBM DB2 Information Integrator Introduction to Replication and Event Publishing*
Set up and administer Q replication and event publishing	*IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference*
Learn about last-minute changes to the product	The Installation Notes on the CD-ROM or the Release Notes that are installed with the products.
Find technical support resources and customer support options	www.ibm.com/software/data/integration/db2ii/
Find classes available from IBM Learning Services	www.ibm.com/services/learning/
Migrate from earlier versions of SQL replication to SQL replication in Version 8	SQL replication product documentation at www.ibm.com/software/data/dpropr/library.html
Tuning performance for Replication in Version 8	DB2 Replication tuning product documentation at www.ibm.com/software/data/dpropr/library.html
Debug error messages	On Linux, UNIX, Windows, z/OS: See the *Message Reference Volume 1*. On OS/400: Press F1 when you receive an error message.
Find out about other DB2 and Information Integrator information	Product Web pages at www.ibm.com/software/data/

What's new for DB2 replication for Version 8?

This section summarizes the major changes made to DB2 replication since Version 7. It introduces what's new in the base Version 8 product, as well as enhancements released in subsequent fix packs. These changes include usability improvements, performance improvements, new function, serviceability improvements, changes to replication system commands, changes to control tables, and functions no longer supported. They are described in detail in the rest of this book.

- "What's new in Version 8.2?"
- "What's new in Version 8.1.4?" on page xvi
- "What's new in Version 8 fix pack 2?" on page xvi
- "What's new in Version 8.1?" on page xvii
- "Functions no longer supported" on page xxv

What's new in Version 8.2?

The following replication enhancements were added in Version 8.2:

New replication solutions

Q replication, available in Version 8.2, offers a new replication solution to take advantage of the power and flexibility of Websphere MQ by replicating over message queues. Many different configurations are possible, from unidirectional to multidirectional replication, including peer-to-peer.

A feature of Q replication, called event publishing, converts source changes to XML messages which are sent over Websphere MQ message queues to either your own or third-party user applications.

This book does not address issues specific to Q replication, but some functions, such as the Replication Alert Monitor, are shared by SQL and Q replication, and are addressed here.

New function

Compare source and target tables: Use the **asntdiff** command for both SQL and Q replication to compare a source table with a target table and generate a list of differences between the two.

Synchronize source and target tables: Use the **asntrep** command for both SQL and Q replication to synchronize a source and target table by repairing differences between the two tables.

Updated monitoring: Monitoring functions have been expanded to include Q replication.

List DB2 replication services: The **asnslist** command for Windows allows you to list the SQL and Q replication services in the Windows Service Control Manager (SCM). You can optionally use the command to list details about each service.

Beginning in fix pack 5, the restriction on including LOB columns in update-anywhere and replica scenarios has been removed, provided that conflict detection is disabled.

What's new in Version 8.1.4?

The following replication enhancements were added in Version 8.1.4:

New function

Support for longer object name lengths in DB2 for z/OS: Replication now supports schema and table names of up to 128 bytes on z/OS when DB2 UDB for z/OS Version 8 is run in new-function mode.

List aliases and user IDs in the password file: The **asnpwd** command allows you to list the aliases and user IDs contained in the password file. You can also use the **encrypt** parameter of the **asnpwd** command to encrypt either all of the entries in a file or just the password entry in a file.

Performance improvements

Improved availability of data on Oracle sources: The Apply program no longer needs to issue lock table statements for CCD tables on Oracle sources. To take advantage of this improvement, you must migrate any existing registrations and subscriptions for Oracle sources following the instructions in the *IBM DB2 Information Integrator Migration Guide: Migrating to SQL Replication Version 8*.

What's new in Version 8 fix pack 2?

The following replication enhancements were added in Version 8 fix pack 2:

Usability improvements

Viewing performance and statistical data: You can view performance and statistical data about the Capture, Apply, and Monitor programs. You can query the data and save it to a file or print it using the Replication Center.

Scheduling replication: In Version 8, you can schedule different times for master-to-replica replication and replica-to-master replication using the Replication Center.

Viewing messages generated by Apply and Monitor programs. You can use the Replication Center to view messages generated by the Apply and Monitor programs (APPLYTRACE and MONTRACE).

Performance improvements

IASP support: On iSeries, you can catalog the database that is available from base Auxiliary Storage Pools (ASP) or from Independent Auxiliary Storage Pools (IASP).

New function

The Capture program can now capture changes from multi-partitioned tables: If you are running DB2 Enterprise Server Edition, you can capture changes to source tables that are spread across multiple partition tables.

Full refresh of single members: You can add one or more members to an existing subscription set without performing a full refresh for all members. You can also disable individual members of a subscription set.

Additional historical data in control tables: DB2 replication provides additional historical data in the control tables, which describe replication activities. Three new tables that contain such data are the Apply trace (IBMSNAP_APPLYTRACE) table, the Capture monitor (IBMSNAP_CAPMON) table, and the Monitor trace (IBMSNAP_MONTRACE) table. You can query the data using the Replication Center.

Transactional mode processing is supported: The COMMIT_COUNT column was added to the IBMSNAP_SUBS_SET table to support transactional mode processing in the CCD table. You can also use this column to control how often changes to the data for this target table will be committed.

Changes to control tables

The IBMSNAP_PARTITIONINFO table was added. It contains information that enables the Capture program to restart from the earliest required log sequence number.

What's new in Version 8.1?

Usability improvements

Enhanced handshake mechanism between the Capture and Apply programs: The handshake is a mechanism that the Apply program uses to tell the Capture program to start capturing data for a replication source. This mechanism has been changed and enhanced for Version 8. The Apply program inserts signals into the new signal (IBMSNAP_SIGNAL) table to control when the Capture program should start capturing data for a source.

Capture and Apply programs can be started in any order: In Version 8, you can start the Capture program after you start the Apply program, or you can start the Apply program after you start the Capture program. In Version 7, you had to start the Capture program before you started the Apply program.

Adding registrations and subscription sets while the Capture program is running: You can register new replication sources, update existing registrations, add new subscription sets, or update existing subscription sets without reinitializing the Capture program or stopping and restarting it.

Greater control over what is captured for each registration: When you register a table for replication you can specify whether you want the Capture program to capture changes for a row whenever *any* column of the table changes or only when a *registered* column changes. In previous versions, you could control what was captured using a start-up parameter for the Capture program, which meant all tables were treated the same. The start-up parameter is not available in V8 because you can control what is captured for each registration.

Greater control over recapturing data from replicas: When you register a source, you can specify if you want changes recaptured from some tables but not others. By default:

- Changes are not recaptured from replica tables and forwarded to other replica tables.

- Changes to master tables in update-anywhere replication are recaptured and sent to replica tables.

One Windows service per program: In Version 7, you could create only one Windows service to operate all of your Capture and Apply programs. Now you can create *separate* services for each Capture and Apply program, as well as the Replication Alert Monitor. You can use each service to start or stop replication. You can use either the Replication Center or new commands to create (**asnscrt** command) or drop (**asnsdrop** command) a service for replication programs.

ARM support for the Capture and Apply programs: For the z/OS environment, the Capture and Apply programs, and the Replication Alert Monitor, are enabled for the MVS Automatic Restart Manager (ARM). The ARM is an MVS recovery function that can improve the availability of specific batch jobs or started tasks. When a job or task fails, or the system on which it is running fails, the ARM can restart the job or task without operator intervention. The ARM uses element names to identify the applications with which it works, and each ARM-enabled application uses a unique element name that it uses to communicate with the ARM. The element names for replication are: ASNTC*xxxxyyyy* for the Capture program, ASNTA*xxxxyyyy* for the Apply program, and ASNAM*xxxxyyyy* for the Replication Alert Monitor.

Improved messages: Existing messages were improved and new messages were added. The explanation and user response sections were updated.

Performance improvements

Fewer joins between replication tables: In Version 8, joins have been eliminated in some situations. The Apply program does not need to join the CD and UOW table to populate user copy target tables under many circumstances. Also, the CD and UOW tables do not need to be joined for pruning.

Capture pruning runs concurrently with reading the DB2 log (UNIX, Windows, z/OS): The Capture program reads the DB2 log while it prunes tables; therefore, pruning does not affect capture latency. In Version 7, the Capture program performed these tasks serially, not concurrently. Also, in Version 8, the Capture program prunes the UOW table, CD tables, trace tables, as well as the new signal (IBMSNAP_SIGNAL) table and monitor (IBMSNAP_CAPMON) table.

Faster full refreshes of target tables (UNIX, Windows, z/OS): DB2 replication takes advantage of the improvements to the load utility in the following DB2 products to provide faster full refreshes of target tables:

- DB2 Universal Database for Windows and UNIX, Version 8
- DB2 Universal Database for z/OS and OS/390, Version 7 or later

Apply program optimizes processing if it has only one subscription set: In Version 8, you can start the Apply program so that it will cache and reuse information about a single subscription set. Using the new **opt4one** keyword improves CPU usage and throughput rates.

Fewer updates for subscription sets with multiple members: Compared to previous versions of DB2 replication, in Version 8 the Apply program makes fewer updates to control tables for subscription sets with multiple members.

New user interface

With Version 8, you can set up and maintain your replication environment, operate the Capture and Apply programs, and the Replication Alert Monitor using one administration tool. The new DB2 Replication Center is a graphical tool that supports administration for DB2-to-DB2 replication environments and administration for replication between DB2 and non-DB2 relational databases.

The Replication Center is part of the DB2 Control Center set of tools and has the look and feel of the other DB2 centers. The Replication Center includes all of the replication function previously available from the DB2 Control Center and the DB2 DataJoiner Replication Administration (DJRA) tool. The Replication Center also has a launchpad that helps you perform the basic functions needed to set up a DB2 replication environment. The launchpad shows you graphically how the different steps are related to one another.

You can use the Replication Center to:

- Define defaults in profiles for creating control tables, source objects, and target objects
- Create replication control tables
- Register replication sources
- Create subscription sets and add subscription-set members to subscription sets
- Operate the Capture program
- Operate the Apply program
- Monitor the replication process
- Perform basic troubleshooting for replication
- Specify the LOADX option
- View messages generated by the Apply and Monitor programs (APPLYTRACE and MONTRACE)
- View performance and statistical data

You can also use the Replication Center to perform many other replication administration tasks.

New function

Multiple Capture programs can concurrently read from the same DB2 log or journal: You can run more than one Capture program against a single DB2 log (DB2 catalog) or journal. For z/OS data-sharing groups, multiple Capture programs can read from the logs for the data-sharing group. Each Capture program is independent of any others. If necessary, you can register a single source table to more than one Capture program. Therefore, if you have low latency tables, they can have a dedicated Capture program so that they have different run-time priority and different Capture characteristics (such as pruning interval). Or, different organizations can maintain their own replication environments using the same source data, but different Capture programs. On z/OS operating systems, you can use multiple Capture programs to support a mixture of ASCII, EBCDIC, and UNICODE source tables within a single DB2 subsystem.

Multiple non-DB2 relational sources per federated database: If your replication environment includes non-DB2 sources, you can define multiple non-DB2 relational sources in a single federated database.

Automated monitoring: The new Replication Alert Monitor runs continuously and monitors the Capture and Apply programs for you. You define thresholds for

criteria that you want to monitor, and specify people who should be contacted automatically via e-mail when those thresholds are met or exceeded. You can use the Replication Center or two new commands (**asnmon** and **asnmcmd**) to configure and operate the Replication Alert Monitor.

On-demand monitoring: You can query the status of the Capture, Apply, and Monitor programs using the **asnccmd**, **asnacmd**, **asnmcmd** status commands.

Encrypted password file (UNIX, Windows): In Version 7, the password file used by the Apply program and the Replication Analyzer contained plain text, not encrypted information. In Version 8, the passwords in the password file are encrypted. No passwords are stored in plain text. A new command (**asnpwd**) enables you to create and maintain the password file.

Improved ASNLOAD exit routine (UNIX, Windows, z/OS): The ASNLOAD exit routine is shipped as a sample exit routine in both source format (C) and compiled format. The sample exit routine differs on each DB2 platform, in each case taking advantage of the utility options offered on the platform. You can use the sample compiled program exit routine as provided and you can influence the behavior in some cases by customizing the replication configuration, or you can customize the exit routine code itself.

More control over cold starts (UNIX, Windows, and z/OS): The **warm** start-up parameter is replaced by two parameters to give you more control over cold starts:

warmsi
: If warm-start information is available, the Capture program resumes processing where it ended in its previous run. If this is the first time that the Capture program is starting or the new restart (IBMSNAP_RESTART) table is empty, the Capture program switches to cold start. This is the default start-up parameter in Version 8.

warmsa
: If warm-start information is available, the Capture program resumes processing where it ended in its previous run. If the Capture program cannot warm start, it switches to a cold start.

More frequent commits by the Apply program: In many situations, if you have user-copy, point-in-time, CCD, or replica target tables in a subscription set, you can specify that you want the Apply program to commit its work after it processes a specified number of transactions. To do so, you must run the Apply program in transaction mode.

Referential integrity for more types of target tables: In many situations, you can have referential integrity on user-copy and point-in-time target tables by starting the Apply program so that it commits its work in transaction mode.

More ways to set operational parameters for the Capture program: You can use the shipped defaults to operate the Capture program or you can create new defaults using the Capture parameters (IBMSNAP_CAPPARMS) table to suit your replication environment. You can also supply operational parameters for the Capture program when you start the program, if you do not want to use the defaults for that session. While the Capture program is running, you can change the operational parameters using the Replication Center, the **chgparms** keyword of the **asnccmd** command (UNIX, Windows, z/OS), or the **OVRDPRCAPA** command (iSeries). These changes last until you end the session or until you issue another change command.

New option for replicating changes to target-key columns: In Version 7, you could ensure that changes to key columns were replicated properly to your target tables by registering your source table to capture updates as delete/insert pairs. In Version 8, when you define a subscription set member, you can specify whether the Apply program should use the before-image values or the after-image values when the Apply program builds a WHERE clause using the primary-key columns in its predicates. By using before-image values, you can avoid the conversion of an update to an insert. You can specify either that a registration use delete/insert pairs for updates or that the subscription-set member use before-image values in Apply WHERE-clause predicates.

More tables pruned by the Capture program: The Capture program prunes the following tables: CD tables, UOW table, trace (IBMSNAP_CAPTRACE) table, as well as the new signal (IBMSNAP_SIGNAL) table and monitor (IBMSNAP_CAPMON) table.

Longer table names and column names: DB2 replication now supports source table and target table names up to 128 characters, and column names up to 30 characters for databases that support long names.

Adding columns to source and CD tables while the Capture program is running: You can add columns to your replication source tables without reinitializing the Capture program or stopping and restarting it. On UNIX, Windows, and z/OS, you can also alter the CD table while the Capture program is running.

New signals to control the Capture program: The Capture program can now be controlled by signals written to the signal (IBMSNAP_SIGNAL) table. The signal table provides a way to communicate with the Capture program through log records. Capture uses the signals for the following situations:

- To determine when to start capturing changes for a particular table
- To determine when to terminate
- Whether it must perform update-anywhere replication
- To provide the log sequence number for setting a precise end point for Apply events

Not only does the signal table let the Apply program tell the Capture program when to start capturing data, it also allows for precise termination of log record reading and for user-defined signals through log records.

Replicating Data Links values (AIX, Solaris Operating Environment, Windows, iSeries):

- If you have a DATALINK value pointing to an external file, you can retrieve consistent versions of files if the column is defined with RECOVERY YES. In past releases, DB2 would replicate the latest copy of the file and could not guarantee that the file being replicated was consistent with the replicated database data values.
- You can maintain the same target file across multiple changes in the source database.
- For the AIX and Windows operating systems, and the Solaris operating environment, you can connect to the DB2 Data Links Manager replication daemon (DLFM_ASNCOPYD) to retrieve and store Data Links files for replication. You do not need to start and maintain a separate ASNDLCOPYD daemon as in previous releases. On OS/400, you still need to start and maintain a separate ASNDLCOPYD daemon.

Unicode encoding schemes added (z/OS): DB2 DataPropagator for z/OS Version 8 supports UNICODE and ASCII encoding schemes. This function was introduced in DB2 DataPropagator for OS/390 Version 7.

64-bit support added (Windows, UNIX, z/OS): In Version 8, you can replicate on operating systems where DB2 offers 64-bit support. Applications running on 64-bit operating systems benefit from the increased memory address space that these systems provide.

Migration utility: The new replication migration utility (**asnmig8**) consists of a set of migration scripts that you can use to convert all Version 5, Version 6, or Version 7 replication tables to Version 8 formats.

Serviceability improvements

New trace facility (UNIX, Windows, z/OS): The new replication trace facility (**asntrc**) is similar to the DB2 trace facilities. You can start or stop the trace facility without stopping and restarting the Capture and Apply programs. Also, the trace output is compact, which usually results in smaller trace files than were generated in previous releases, and is consistent with the DB2 trace format.

Replication Analyzer program updated: The Replication Analyzer program was modified to analyze the new V8 features. The Analyzer generates reports about the state of the replication control tables on the specified systems. These reports can be used to verify and tune your replication environment or to diagnose problems. You can download the Analyzer and its documentation from the Web.

New and updated error messages: New error messages were added for new functionality. Existing messages were updated to improve readability.

Changes to replication system commands

New and changed replication system commands for UNIX, Windows, z/OS: The syntax of existing system commands on Windows, UNIX, and z/OS was modified. The following changes were also made:

- The Capture command line (**asncmd**) was renamed to **asnccmd** so that it is consistent with the new Apply command line (**asnacmd**), which you use to operate the Apply program, and the new Monitor command line (**asnmcmd**), which you use to operate the Monitor program.
- The **asnccp** command for starting the Capture program was renamed to **asncap**.

The following new system commands, which run on UNIX, Windows, and z/OS operating systems, were added:

- **asnacmd** (Apply command line) operates and stops the Apply program.
- **asnmon** (Monitor command) starts the Replication Alert Monitor
- **asnmcmd** (Monitoring command line) operates and stops the Replication Alert Monitor.
- **asnanalyze** (Analyzer command) generates reports about the state of the replication control tables.
- **asnpwd** (Password command) creates and maintains password files needed in a distributed replication environment.
- **asntrc** (Trace facility) replaces the startup options to generate a trace for the Capture and Apply programs.

New and changed replication system commands for OS/400 operating systems (iSeries): The following new system commands, which run on an OS/400 system, were added:

- **ADDDPRREG** (Add a DPR registration) registers a user table for replication.
- **RMVDPRREG** (Remove a DPR registration) removes a user table from the list of source tables available for replication.
- **ADDDPRSUB** (Add DPR subscription set) creates an empty subscription set or a subscription set with one member.
- **RMVDPRSUB** (Remove DPR subscription set) removes an empty set or a set and all of its members.
- **ADDDPRSUBM** (Add DPR subscription-set member) adds a member to an existing subscription set.
- **RMVDPRSUBM** (Remove DPR subscription-set member) removes a single subscription-set member from a subscription set.
- **OVRDPRCAPA** (Override DPR Capture attributes) changes the attributes for the Capture program that is currently running.
- **ANZDPR** (Analyzer) generates reports about the state of the replication control tables on the specified systems. These reports can be used to verify and tune your replication environment or to diagnose problems.
- **WRKDPRTRC** (Trace options) operates various trace options such as Dump.

Some changes were made to existing system commands for OS/400 systems:

- **DPRVSN** (DataPropagator Version) parameter was removed from all system commands.
- **CAPCTLLIB** (Capture Control Library) parameter was added to the Capture commands.
- New parameters were added to the **CHGDPRCAPA** (Change DPR Capture attributes) and **STRDPRCAP** (Start DPR Capture) commands to take advantage of the new tracing and monitoring functions.
- A new parameter was added to the **ENDDPRCAP** (End DPR Capture) command so that it automatically reorganizes the CD and UOW tables to reclaim space.
- New parameters were added to the **STRDPRAPY** (Start DPR Apply) command that enable the Apply program to run only once, clean up the Apply trail (IBMSNAP_APPLYTRAIL) table, and optimize processing of single subscription sets.

Changes to control tables

Substantial changes were made to the control table structures in Version 8 to support new function and to improve usability. New tables were added, some existing tables were changed, and a few tables were made obsolete by new tables.

The following new tables were added:

- IBMSNAP_APPENQ ensures that only one Apply program is running for a single Apply qualifier.
- IBMSNAP_APPLYTRACE contains important messages from the Apply program.
- IBMSNAP_APPPARMS contains parameters that you can modify to control the operation of the Apply program.
- IBMSNAP_CAPENQ ensures that only one Capture program is running for a single Capture schema.

- IBMSNAP_CAPMON contains operational statistics for monitoring the progress of the Capture program.
- IBMSNAP_CAPSCHEMAS contains the names of all Capture schemas.
- IBMSNAP_PRUNE_SET coordinates the pruning of CD tables.
- IBMSNAP_RESTART enables the Capture program to resume capturing from the correct point in the log or journal.
- IBMSNAP_SIGNAL contains signals used to control the Capture program.

The following new tables were added for the Replication Alert Monitor:

- IBMSNAP_ALERTS contains a history of all alerts issued by the Replication Alert Monitor.
- IBMSNAP_CONDITIONS contains alert conditions for each monitored server.
- IBMSNAP_CONTACTGRP maps contacts with groups.
- IBMSNAP_CONTACTS contains contact names and addresses.
- IBMSNAP_GROUPS contains contact groups.
- IBMSNAP_MONENQ ensures that only one Monitor process is running for a single Monitor qualifier.
- IBMSNAP_MONPARMS contains parameters that you can modify to control the operation of the Replication Alert Monitor program.
- IBMSNAP_MONSERVERS contains the most recent time that the Replication Alert Monitor monitored a Capture or Apply control server.
- IBMSNAP_MONTRACE traces Replication Alert Monitor activity.
- IBMSNAP_MONTRAIL contains a history of Monitor activity for every Monitor cycle.

The following tables were changed:

- IBMSNAP_APPLYTRAIL
- IBMSNAP_AUTHTKN (OS/400 only)
- IBMSNAP_CAPPARMS (formerly known as IBMSNAP_CCPPARMS)
- IBMSNAP_CAPTRACE (formerly known as IBMSNAP_TRACE)
- IBMSNAP_PRUNCNTL
- IBMSNAP_REG_EXT (OS/400 only)
- IBMSNAP_REGISTER
- IBMSNAP_SUBS_COLS
- IBMSNAP_SUBS_EVENT
- IBMSNAP_SUBS_MEMBR
- IBMSNAP_SUBS_SET
- IBMSNAP_UOW

CD tables were also changed.

The following tables from previous versions of DB2 replication are now obsolete:

- IBMSNAP_CRITSEC is replaced by IBMSNAP_SIGNAL.
- IBMSNAP_WARMSTART is replaced by IBMSNAP_RESTART.

Functions no longer supported

Windows System Services is no longer supported by the DB2 Replication Center. The Windows commands and the Windows services dialog both allow operational access to Windows services.

The DB2 DataJoiner Replication Administration (DJRA) tool is not supported for Version 8. You cannot use DJRA to create Version 8 replication control tables, and you cannot use DJRA to register sources or define subscription sets that use V8 control tables. DJRA continues to be supported for Version 7 replication environments. Use the Replication Center for V8 replication environments.

The DB2 Control Center does not support Version 8 replication control tables, and you cannot use the Control Center to register sources or define subscription sets that use V8 control tables. You can use the Control Center for Version 7 replication environments. Use the Replication Center for V8 replication environments.

The **ASNSAT** command is no longer available. Also, the ability to generalize replication subscriptions and set up a DB2 satellite replication environment is no longer available from the Satellite Administration Center. If you require data replication for a mobile work force, consider migrating your satellite DB2 databases to DB2 Everyplace, Version 8. For additional information, contact your IBM representative.

Part 1. Replication guide

This part of the book contains the following chapters:

Chapter 1, “Planning for SQL replication,” on page 3 describes how to plan your replication environment.

Chapter 2, “Configuring servers for SQL replication,” on page 15 describes how to prepare your environment for replication.

Chapter 3, “Registering tables and views as SQL replication sources,” on page 35 describes what you need to know to register replication sources.

Chapter 4, “Subscribing to sources for SQL replication,” on page 57 describes what you need to know to create subscription sets and add members to subscription sets.

Chapter 5, “Replicating special data types in SQL replication,” on page 85 describes the replication options for LOB and DATALINK values in source tables.

Chapter 6, “Subsetting data in an SQL replication environment,” on page 93 describes how to customize what data is captured and applied to the target as well as how the data is applied to the target.

Chapter 7, “Manipulating data in an SQL replication environment,” on page 97 describes how to use the Capture program or the Apply program to manipulate the source data.

Chapter 8, “Customizing and running replication SQL scripts for SQL replication,” on page 101 describes how to run SQL in your replication environment.

Chapter 9, “Operating the Capture program for SQL replication,” on page 103 describes how to operate the Capture program for all operating-system environments.

Chapter 10, “Operating the Apply program for SQL replication,” on page 121 describes how to operate the Apply program for all operating-system environments.

Chapter 11, “Monitoring replication with the Replication Alert Monitor,” on page 143 describes how to use the Replication Alert Monitor to monitor your replication environment.

Chapter 12, “On-demand reporting for SQL replication,” on page 165 describes how to generate and view reports about your replication environment on demand.

Chapter 13, “Making changes to an SQL replication environment,” on page 173 describes how to change your replication environment.

Chapter 14, “Maintaining an SQL replication environment,” on page 205 describes how to maintain your source tables, control tables, and target tables.

Chapter 1. Planning for SQL replication

This chapter describes how to plan your replication environment. It contains the following sections:

- "Migration planning"
- "Memory planning"
- "Storage planning" on page 5
- "Planning for conflict detection" on page 10
- "Planning for non-DB2 relational sources" on page 10
- "Planning for code page translation" on page 11
- "Replication planning for DB2 UDB for z/OS" on page 13
- "Performance tuning" on page 13

Migration planning

If you are migrating from an existing replication environment, certain migration issues need to be considered. The *Migration Guide: Migrating to DB2 Replication* describes how to migrate from an existing DB2 replication environment to Version 8 replication. It also describes how to migrate replication environments that currently use DB2 DataJoiner® to replicate data to or from non-DB2 relational servers. This document is available online at www.ibm.com/software/data/dpropr/library.html

Memory planning

You must plan for the amount of memory required by DB2 replication. DB2 replication uses memory only as needed. The amount of memory required is directly proportional to how much data is being replicated from the source and the concurrency of the transactions. Basically, the more data that is being replicated and the more concurrent transactions you have, the more memory is required by replication.

Running the Capture and Apply programs can consume a significant amount of memory resources.

Memory used by the Capture program

When the Capture program reads the DB2 log, the Capture program stores individual transaction records in memory until it reads the associated commit or abort record. Data associated with an aborted transaction is cleared from memory, and data associated with a commit record is written to the CD table and the UOW table. The committed transactions stay in memory until the Capture program commits its work when it reaches its commit interval.

To monitor how much memory the Capture program is using, look in the CURRENT_MEMORY column of the Capture monitor (IBMSNAP_CAPMON) table.

You can set the **memory_limit** parameter when you start the Capture program to ensure that Capture uses a specified amount of memory for storage that is associated with transactions. Other storage use is not limited by this parameter.

You can also change the **memory_limit** parameter while the Capture program is running. If Capture reaches the memory limit, it writes some transactions to a spill file. See "Planning space requirements for spill files for the Capture program" on page 8 for the storage requirements of spill files. You need to consider the memory resources that are used by the Capture program in relation to its storage space requirements.

You should also consider the size of user transactions and the commit interval when planning for the Capture program's memory requirements. Long running batch jobs without interim commits take a lot of memory when you run the Capture program. Generally, the smaller the commit interval, the less memory required by the Capture program.

Reading information about registrations: Information about active registrations is read and stored in memory when you start an instance of the Capture program and when you add registrations dynamically while the Capture program is running.

Reading log records (Linux, UNIX, Windows, z/OS): When DB2 replication reads log records it uses a memory buffer. The default size of the buffer on Linux, UNIX and Windows operating systems is fifty 4 KB pages. The default size on the z/OS operating system is sixty-six 1 KB pages, and it is ECSA (extended common service area) storage. Replication uses ECSA only in this situation.

Memory used on OS/400: CURRENT_MEMORY is the up-to-date account of extra memory allocated for holding the transaction records beyond the memory used by standard I/O buffers for the active CD tables. It is an indication of how much extra memory is being used to hold the large number of transactions. It is not an accurate sum of all the memory used by the specific journal job.

Information stored in the Capture monitor (IBMSNAP_CAPMON) table provides operational statistics to help you tune memory usage. Note that the values in this table are for a particular Capture monitor interval, they are not cumulative across monitor intervals. The data in the CURRENT_MEMORY column does not contain an additive count. It reflects the memory in use at the end of the monitor interval when the record is created. The Capture monitor interval determines how frequently the Capture program inserts data into this table. Use one of the following methods to tune the amount of memory being used by the Capture program:

Tuning memory limit to allow for spills:

1. When you start the Capture program, use the default memory limit.
2. Check if data spilled from memory to a temporary file by looking at the TRANS_SPILLED column in the Capture monitor (IBMSNAP_CAPMON) table. This column shows the number of source system transactions that spilled to disk due to memory restrictions during a particular Capture monitor interval.
3. If data spilled from memory, either use a higher memory limit or a lower commit interval.

Tuning memory limit to prevent spills:

1. When you start the Capture program, set a high memory limit. (How high depends on your system resources.)
2. Check how much memory is being used by looking at the CURRENT_MEMORY column in the Capture monitor (IBMSNAP_CAPMON)

table. This column shows the amount of memory (in bytes) that the Capture program used during a particular Capture monitor interval.

3. If much less memory is being used than what you specified for the memory limit, set a lower value for the memory limit.

Memory used by the Apply program

When the Apply program fetches data, it typically uses a small amount of memory for fetching individual rows. The amount of memory used is proportional to the size of the table columns and the number of rows fetched at one time. For example, if the Apply program is fetching a LOB column, it could potentially use 2 GB of memory.

Information about active subscription sets is read and stored in memory when the Apply program is running. The amount of memory used at one time by the Apply program is generally proportional to the amount of memory required to process the subscription set that has the most members.

Memory used by the Replication Alert Monitor

Memory is used for storing the definitions and for keeping the alerts in memory before they are sent as notifications. The amount of memory needed for the definitions is directly proportional to the number of definitions. The Replication Alert Monitor reserves 32 KB of memory for storing alert notifications. More memory is requested, as needed, and released when no longer required.

Storage planning

In addition to the storage required for DB2, you must ensure that storage is available for replication for the following items:

Database log and journal data
The additional data logged to support the replication of data. See "Planning log impact" on page 6 for details.

Target tables and control tables
The replicated data and control tables (including CD tables). See "Planning the storage requirements of target tables and control tables" on page 7 for details.

Temporary files
The data stored by replication programs in spill files and diagnostic log files (for example, *CAP.log and *APP.log). See "Planning storage requirements for temporary files" on page 8 for details.

OS/400: Current receiver size for Capture
For registered source tables yet to be captured, the journal entries must remain on the current chain of receivers.For more information, see "Using the delete journal receiver exit routine" on page 33.

All of the sizes given in the following sections are estimates only. To prepare and design a production-ready system, you must also account for such things as failure prevention. For example, the holding period of data (discussed in "Planning the storage requirements of target tables and control tables" on page 7) might need to be increased to account for potential network outages.

Tip: If storage estimates seem unreasonably high, reexamine how frequently the Apply program runs subscription sets and how frequently your replication tables are pruned. You must consider trade-offs between storage usage, capacity for failure tolerance, and CPU overhead.

Planning log impact

You must plan the log impact for the replication servers. DB2 replication requires that both the source and the target tables be logged (journaled).

Planning the log impact for DB2 source servers

In general you need an additional three times the current log volume for all tables involved in replication. Basically, you need log space for the source table as well as the CD table and the replication control tables. This section provides other factors that can help you make a more accurate estimate of the log impact that you can expect in your replication environment.

Consider the updates made to the source database by your applications and the replication requirements. For example, if an updating application typically updates 60% of the columns in a table, the replication requirements could cause the log records to grow by more than half compared to a similar table that is not replicated.

Linux, UNIX, Windows, and z/OS:

- DB2 logs full-row images for each UPDATE statement. This occurs because, before you can replicate a table, you must create it (or alter it) with the DATA CAPTURE CHANGES keywords.
- One of the replication requirements that adds the most to the log is the capturing of before- and after-images (as for replica target tables in update-anywhere replication scenarios). One way to reduce the log volume is to reduce the number of columns defined for the replication source. For example, do not capture before-images if they're not required.

OS/400:

- DB2 logs full-row images for each UPDATE statement. One way to reduce the log volume is to reduce the number of columns defined for the replication source, for example, do not capture before-images if they're not required.
- To minimize the amount of storage used for CD tables and UOW tables, frequently reorganize these tables because pruning does not recover DASD for you. You can use the keyword RGZCTLTBL (Reorganize control tables) on the **ENDDPRCAP** command to reorganize control tables. Observe the DASD usage patterns under normal operating conditions to help you predict and manage DASD usage. If journaling is on, also take into account that the log or journal volume increases as DB2 log insertions to and deletions from the UOW table and CD tables.
- When the current receiver is full, the system switches to a new one; you can optionally save and delete old ones no longer needed for replication. When a system handles a large number of transactions, the Capture program can occasionally lag behind. If Capture is frequently lagging behind, you can separate your source tables into multiple journals to distribute the workload to multiple instances of the Capture program.

Planning the log impact for target servers

In addition to logging for the source database, there is also logging for the target database, where the rows are applied. The impact to the log depends on the commit mode that you choose for the Apply program.

Table mode

In table-mode processing, the Apply program issues a single commit after all fetched data is applied. The Apply program does not issue interim checkpoints. In this case, you should estimate the maximum amount of data that the Apply program will process in one time interval and adjust the log space to accommodate that amount of data.

Transaction mode

In transaction-mode processing, the Apply program copies every update in the source transaction order to the target tables and commits these changes on a transaction boundary at an interval. You set the interval for the interim commits by setting the value of *x* in the subscription set option **commit_count**(*x*). After the Apply program fetches all answer sets, it applies the contents of the spill files in the order of commit sequence. This type of processing allows all spill files to be open and processed at the same time. For example, if you set commit count to 1, the Apply program commits after each transaction, if you set commit count to 2, it commits after each set of two transactions.

OS/400: If the target operating system is OS/400, you also need to consider the log space (journal receivers space) of the target tables. Because journal receivers for target tables on OS/400 can be created with the MNGRCV(*SYSTEM) and DLTRCV(*YES) parameters, and because you need to journal only the after-image columns, use the following formula to estimate the volume of the journal receivers for the target tables:

```
journal_receiver_volume=target_table_row_length X journal_receiver_threshold
```

Planning the storage requirements of target tables and control tables

You must estimate the volume of new target tables. The space required for a target table is usually no greater than that of the source table, but can be much larger if the target table is denormalized or includes before-images (in addition to after-images) or history data. Target table size depends on what you choose to replicate, for example, the percentage of the source table you are replicating, the data type of columns you're replicating, whether you're replicating before- and after-images, whether you're adding computed columns, whether you're subsetting rows, whether any transformations are performed during replication.

The CD tables and some replication control tables (IBMSNAP_UOW, IBMSNAP_CAPTRACE, IBMSNAP_APPLYTRACE, IBMSNAP_APPLYTRAIL, IBMSNAP_CAPMON, IBMSNAP_ALERTS) also affect the disk space required for DB2 source databases. These tables can grow very large depending on how you set up your replication environment. The space required for the other replication control tables is generally small and static.

The CD tables grow in size for every change made to a source table until the Capture program prunes the CD table. To estimate the space required for the CD tables, first determine how long you want to keep the data before pruning it, then specify how often the Capture program should automatically prune these tables or how often you will prune the tables using a command.

When calculating the number of bytes of data replicated, you need to include 21 bytes for overhead data for each row that is added to the CD tables by the Capture program. Determine the period of time for which the Capture program should be able to keep capturing data into CD tables, even when the data cannot be applied - for example, in the case of a network outage. Estimate the number of inserts, updates, and deletes that typically would be captured for the source table within that contingency time period.

To determine the recommended size for the CD table, use the following guideline:

```
recommended_CD_size =
  ( (21 bytes) + sum(length of all registered columns) ) X
  (number of inserts, updates, and deletes to source table
   during the contingency period)
```

Example: If the rows in the CD table are 100 bytes long (plus the 21 bytes for overhead), and 100,000 updates are captured during a 24-hour contingency period, the storage required for the CD table is about 12 MB.

Registered columns in this formula include both before- and after-image columns. If updates are being converted to pairs of INSERT and DELETE operations, then take them into account when determining the total number of inserts, updates, and deletes. For example, count each update to the source table as two rows in the CD table.

The UOW table grows and shrinks based on the number of rows inserted by the Capture program during a particular commit interval and on the number of rows that are pruned. A row is inserted in the UOW table each time an application transaction issues a COMMIT and the transaction executed an INSERT, DELETE, or UPDATE operation against a registered replication source table. You should initially over-estimate the space required by the table and monitor the space actually used to determine if any space can be recovered.

Planning storage requirements for temporary files

You must plan for the storage requirements of spill files and diagnostic log files.

Planning space requirements for diagnostic log files (Linux, UNIX, Windows, z/OS)

Diagnostic log files store information about the activities of Replication programs, such as when the program started and stopped, and other informational or error messages from the program. By default, the program appends messages to its log file, even after the program is restarted. Ensure that the directories that contain these log files have enough space to store the files. The location of these files depends on the value that you set for the **capture_path**, **apply_path**, and **monitor_path** start-up parameters when you started the Capture program, Apply program, and Replication Alert monitor program, respectively.

If you are concerned about storage, you have the option of reusing the program logs so that each time the program starts it deletes its log and recreates it. You can specify if you want to reuse the log when you start the program.

Planning space requirements for spill files for the Capture program

If the Capture program does not have sufficient memory, it writes (or spills) transactions to spill files. The Capture program writes the biggest transaction to file; however, the biggest transaction is not necessarily the one that exceeded the memory limit.

- **Linux, UNIX, Windows**: On Linux, UNIX and Windows, spill files are always on disk. One file per transaction is created in the **capture_path** directory.
- **OS/400**: On OS/400, spill files are created in library QTEMP, one spill file for each registration that needs a spill file.
- **z/OS**: On z/OS, spill files go to virtual I/O (VIO).

The size of the Capture spill files depends on the following factors:

Memory limit
Use the **memory_limit** operational parameter to specify how much memory can be used by the Capture program. The more memory you allow, the less likely the Capture program will spill to files.

Size of transactions
Larger transactions might increase the need to spill to file.

Number of concurrent transactions
If the Capture program processes more transactions at the same time, or processes interleaved transactions, the Capture program needs to store more information in memory or on disk.

Commit interval
Typically the lower the commit interval the lower the need for storage because Capture has to store information in memory for a shorter period of time before committing it.

Planning space requirements for spill files for the Apply program

The Apply program requires temporary space to store data. (If you are using the ASNLOAD utility, you might have a load input file instead of a load spill file.) The Apply program uses spill files to hold the updates until it applies them to the target tables. In general, the spill files are disk files; however, on z/OS operating systems, you can specify that data be spilled to memory. Unless you have virtual memory constraints, store the spill files in virtual memory rather than on disk.

The size of the spill file is proportional to the size of the data selected for replication during each replication interval. Typically the spill file is approximately two times the size of the data. You can estimate the size of the spill file by comparing the frequency interval (or data-blocking value) planned for the Apply program with the volume of changes in that same time period (or in a peak period of change).

On OS/400, the spill file's row size is a constant 32 KB.

On Linux, UNIX, Windows, z/OS, the spill file's row size is the *target row* size, including any replication overhead columns. The row size is not in DB2 packed internal format, but is in expanded, interpreted character format (as fetched from the SELECT). The row also includes a row length and null terminators on individual column strings. The following example estimates the size of the spill file that is required for the data selected for replication and it does not take into account the extra space needed for the other data that is stored in the spill file.

Example: If change volume peaks at 12,000 updates per hour and the Apply program frequency is planned for one-hour intervals, the spill file must hold one-hour's worth of updates, or 12,000 updates. If each update represents 100 bytes of data, the spill file will be approximately 1.2 MB at a minimum. Additional space is required for the other data that is stored in the spill file.

Planning for conflict detection

If you use standard or enhanced conflict detection, you must store before-images in the CD (or CCD) tables for the replica target tables. Also, the referential integrity rules are restricted. In peer-to-peer and update-anywhere scenarios, or when the Apply program uses transaction mode processing, you should define referential integrity rules that are in keeping with the source rules.

If you use peer-to-peer replication or update-anywhere replication and you do not want to turn on conflict detection, you should design your application environment to prevent update conflicts. If conflicts cannot occur in your application environment, you can save processing cycles by not using conflict detection.

Use either of the following methods to prevent conflicts in peer-to-peer and update-anywhere replication:

Fragmentation by key
Design your application so that the replication source is updated by replicas for key ranges at specific sites. For example, your New York site can update sales records only for the Eastern United States (using ZIP codes[1] less than or equal to 49999 as the key range), but can read all sales records.

Fragmentation by time
Design your application so that the table can be updated *only* during specific time periods at specific sites. The time periods must be sufficiently separated to allow for the replication of any pending changes to be made to the site that is now becoming the master version. Remember to allow for time changes, such as Daylight Savings Time or Summer Time, and for time-zone differences.

Planning for non-DB2 relational sources

Capture triggers are used instead of the Capture program if you are replicating from non-DB2 relational databases. These triggers capture changed data from a non-DB2 relational source table and commit the changed data into CCD tables. Capture triggers affect your transaction throughput rates and log space requirements. Also, if you have existing triggers in your environment you might need to merge them with the new Capture triggers. For more information, see the following sections:

- "Planning transaction throughput rates for Capture triggers"
- "Planning the log impact for non-DB2 relational source servers" on page 11
- "Planning locks for Oracle source servers" on page 11
- "Planning coexistence of pre-existing triggers with Capture triggers" on page 11

Planning transaction throughput rates for Capture triggers

The transaction workload for your source system will increase; trigger-based change capture has an impact on transaction throughput rates. Capture triggers also increase the response time for the updating transactions. The impact is greatest for those transactions that heavily update application source tables that are to be replicated.

1. United States postal codes.

Planning the log impact for non-DB2 relational source servers

For non-DB2 relational source servers, your source applications will need more active log space because the log volume approximately triples for replicated source tables. Changes are captured by triggers on the source tables and are stored in CCD tables, changed data is written within the same commit scope as the changing source tables, and data is later deleted through a trigger-based pruning mechanism. Therefore, each source INSERT, UPDATE, or DELETE operation becomes an INSERT, UPDATE, or DELETE operation, plus an INSERT operation, plus a DELETE operation. The log volume increases even more if you change updates to pairs of DELETE and INSERT operations.

If you run out of log space and the Capture trigger cannot insert a record into the CCD table, the transaction attempted by the user or application program will not complete successfully.

Planning locks for Oracle source servers

Any application currently updating the Oracle source must finish before the Apply program can start applying data. The Apply program must lock the CCD table so that it can process data and set its synchpoint. The locks on the CCD tables are held only until the Apply program sets its synchpoint, not through the entire Apply cycle. Applications that need to update the source table must wait until the Apply program unlocks the CCD table.

Planning coexistence of pre-existing triggers with Capture triggers

The Capture trigger logic is in the SQL script generated by the Replication Center when you register a source. By default, an INSERT trigger, an UPDATE trigger, and a DELETE trigger are created so that those types of changes (insert, update, delete) can be replicated from the source table. The trigger name consists of the name of the CCD table preceded by a letter describing the type of trigger: I for INSERT, U for UPDATE, D for DELETE. For example, if the CCD table name is `undjr02.ccd001`, the name of the generated DELETE trigger is `undjr02.dccd001`. You must *not* change the names of the triggers that are generated in the script.

If a trigger already exists on the table that you want to register for replication and that trigger has the same name as the one that is in the generated script, you'll receive a warning when the script is generated. Do *not* run the generated script because the RDBMS might overwrite the existing trigger. Determine how you want to merge the pre-existing triggers with the new triggers, and create a script that merges your existing logic with the trigger logic generated by the Replication Center.

If the type of trigger that you want to create already exists on the table that you want to register for replication, and the RDBMS allows only one such trigger per table, you must merge the logic before you run the generated script.

Planning for code page translation

Replication components are database applications that rely on the DB2 databases on various operating systems to handle code page translation of data. They work with data using SQL SELECT, INSERT, UPDATE, and DELETE statements.

Replicating data between databases with compatible code pages

If your replication configuration requires SQL statements and data to go between systems with differing code pages, the underlying DB2 protocols such as DRDA handle code page translation. Also, if data is passed between DB2 and non-DB2 relational databases, DB2 replication relies on the underlying database products to handle any necessary code page translation.

If you plan to replicate data between databases with differing code pages, check the *DB2 Administration Guide* to determine if the code pages you have are compatible. For example, if you are using DB2 for Linux, UNIX or Windows see the section on the conversion of character data.

Once you have verified that your databases have compatible code pages, determine if the databases use code pages differently. For example, assume that one database product allows a different code page for each column in a table while another database product does not allow different code pages per column, it requires the code page to be specified only at the database level. A table with multiple code pages in the first product cannot be replicated to a single database in the second product. Therefore, how the databases handle code pages affects how you must set up replication to ensure that data is successfully replicated between the various databases in your environment.

Configuring national language support (NLS) for replication

The NLS configuration for replication is defined when you set up database connectivity between systems. However, if you are running the Capture program on Linux, UNIX or Windows operating systems, the Capture program must use the same code page as the database from which it is capturing the data. If the Capture program does not use the same code page, you must set a DB2 environment variable or registry variable called DB2CODEPAGE.

Setting the code page variable

DB2 derives the code page for an application from the active environment in which the application is running. Typically, when the DB2CODEPAGE variable is not set, the code page is derived from the language ID that is specified by the operating system. In most situations, this value is correct for the Capture program if you use the default code page when you create your database. However, if you create your database with an explicit code page that is something other than the default code page, you must set the DB2CODEPAGE variable for the Capture program. Otherwise, data might not be translated correctly when the Capture program inserts it into a CD table. The value that you use for the DB2CODEPAGE variable must be the same as what you specify on your CREATE DATABASE statement. Refer to the *DB2 Administration Guide* for information about setting the DB2CODEPAGE variable.

Replicating from a code page

If you are replicating source data with a single-byte character set (SBCS) code page to a target with Unicode UTF-8, some single-byte characters in the source database might be translated by DB2 to two or more bytes in the target database. All single-byte characters whose hexadecimal value is 0x80 to 0xff are translated to their two-byte 1208 equivalent. This means that target columns might need to be larger than source columns, otherwise the Apply program might receive SQL errors from DB2.

Some database products implement code page support differently from others, which can impact your replication configuration. For example, the current DB2 on iSeries (OS/400) allows a code page to be specified at the column level, but DB2 for Linux, UNIX, and Windows allows a code page to be specified only at the database level. Therefore, if you have an OS/400 table with multiple columns using different code pages, those columns cannot be replicated to a single DB2 for Linux, UNIX, and Windows database unless all the code pages are compatible.

Setting the LANG variable

If you are running the Capture and Apply programs on a Linux or UNIX system, you might need to set the LANG environment variable. The Capture and Apply programs use the contents of this environment variable to find their message library for your language. For example, if the LANG environmental variable is set to en_US, the Capture program looks for its English message library in the DB2 instance's /sqllib/msg/en_US subdirectory. If Capture cannot find its message library, all messages written to the Capture trace table (ASN_IBMSNAP_TRACE) are ASN0000S.

Replication planning for DB2 UDB for z/OS

DB2 DataPropagator for z/OS Version 8 supports schema and table names of up to 128 bytes. To take advantage of the long name support:

- Create your Capture, Apply, and Monitor control tables under DB2 UDB for z/OS Version 8 in new-function mode.
- Run the Capture, Apply, and Monitor servers under DB2 UDB for z/OS Version 8 in new-function mode

Restriction: If you want to replicate between DB2 UDB for z/OS new-function mode subsystems and DB2 UDB on Linux, Unix, Windows, or iSeries, you must use schema names that are 30 bytes or shorter. If you use schema names that are longer than 30 characters on DB2 UDB for z/OS Version 8 in new-function mode, you cannot replicate between that platform and DB2 UDB for Linux, UNIX, Windows, or iSeries.

Performance tuning

You will want to tune your replication environment for optimal performance. The *Tuning for Replication Performance* document describes how to tune the major components of a DB2 replication environment for maximum performance. This document is available online at www.ibm.com/software/data/dpropr/library.html.

Chapter 2. Configuring servers for SQL replication

You must set up your environment before you can replicate data.

This chapter contains the following sections:

- "Controlling access to replication servers"
- "Authorizing user IDs for replication" on page 17
- "Storing user IDs and passwords for replication (Linux, UNIX, Windows)" on page 22
- "Setting up the replication control tables" on page 22
- "Setting up the replication programs" on page 25
- "Setting up journals (OS/400)" on page 30

Controlling access to replication servers

In most replication environments, data is distributed across servers. If you have such an environment, you must ensure that the replication programs can connect to all servers. You must have the correct software installed to provide connectivity between servers, and you must configure the connectivity between the servers. If you are replicating to non-DB2 relational databases, you must also configure the federated server and related connectivity.

Connectivity requirements for replication

Any workstation that runs the Apply program, the Replication Center, or the replication commands must be able to connect to the source server, Capture control server, Apply control server, and target server databases.

If you use the Replication Alert Monitor, the workstation on which it runs must be able to connect to the Monitor control server and to any server that it monitors. If you want to use the Replication Center to set up monitoring, ensure that the Replication Center can connect to the Monitor control server.

If your replication design involves staging data at a server that is different from the source database, you must carefully consider the communications between the various servers. Be sure to limit the layers of emulation, LAN bridges, and router links required, because these can all affect replication performance.

When the databases are connected to a network, connectivity varies according to the operating systems being connected.

Connecting to non-DB2 relational servers

If you want to replicate data to or from a non-DB2 relational server, you must be able to access the non-DB2 relational server and connect to it.

Before you attempt to replicate from non-DB2 relational source servers, you must set up your federated server and database. There are three main setup steps:

1. Define a wrapper so that the DB2 database can access other non-DB2 relational databases.
2. Define a non-DB2 relational database using a server mapping.

3. If the user ID and password combination that is used to connect to the DB2 database differs from the one used to access the non-DB2 relational database, you must create a user mapping.

Follow the instructions in *DB2 Federated Systems Guide*, GC27–1224, to ensure that your environment is correctly configured.

Connecting to z/OS or iSeries servers from Linux, UNIX, or Windows servers

Ensure that you can connect to all remote servers. To configure connections between z/OS or OS/400 systems and Windows, Linux, or UNIX systems, refer to the *DB2 Connect Quick Beginnings*.

Prerequisites:

The following conditions must exist before you can connect to an iSeries server:

- You must have a DB2 Universal Database or DB2 Connect installed on your workstation.
- You must have TCP/IP set up on your workstation.

Procedure (for iSeries):

To connect to an iSeries server from a DB2 for Windows workstation:

1. Log on to the iSeries server and locate the relational database:
   a. Log on to the iSeries server to which you want to connect.
   b. Submit a **dsprdbdire** command, then specify `local` for *LOCAL.
   c. Locate the name of the relational database in the output. For example, in the following output, the database is called DB2400E:

```
MYDBOS2             9.112.14.67
RCHASDPD            RCHASDPD
DB2400E             *LOCAL
RCHASLJN            RCHASLJN
```

2. Catalog the OS/400 database in DB2 for Windows:
   a. From your Windows workstation, click **Start → Programs → IBM DB2 → Command Window**. The DB2 CLP command window opens.
   b. In the command window, type the following three commands in exact order:

```
db2 catalog tcpip node server_name remote server_name server 446 system
server_name ostype OS400

db2 catalog dcs database rdb_name AS rdb_name

db2 catalog database rdb_name AS rdb_name at node server_name
authentication dcs
```

   Where *server_name* is the TCP/IP host name of the iSeries system, and *rdb_name* is the name of the iSeries relational database that you found in Step 1.
3. In the command window, issue the following command:

```
db2 terminate
```

4. Ensure that the iSeries user profile that you will use to log on to your iSeries system uses CCSID37:
   a. Log on to the iSeries system.
   b. Type the following command, where *user* is the user profile:

```
CHGUSRPRF USRPRF (user) CCSID(37)
```

c. Make sure that the DDM server is started on the iSeries system type:

```
STRTCPSVR SERVER(*DDM)
```

5. Make sure that DB2 for Windows and DB2 for iSeries are connected:

```
db2 connect to rdb_name user user_name using password
```

Authorizing user IDs for replication

If you have to access data in DB2 and non-DB2 relational servers, ensure that the following authorization requirements are met:

- "Authorization requirements for administration"
- "Authorization requirements for the Capture program" on page 18
- "Authorization requirements for Capture triggers on non-DB2 relational databases" on page 19
- "Authorization requirements for the Apply program" on page 19
- "Authorization requirements for the Replication Alert Monitor" on page 21

Authorization requirements for administration

You use the Replication Center to administer replication (see Chapter 15, "Using the Replication Center for SQL replication," on page 219 for details). If your replication environment is only on the OS/400 operating system, you can use the OS/400 system commands to administer replication (see Chapter 19, "System commands for SQL replication (OS/400)," on page 319 for details). To administer replication, you must have at least one user ID on all databases involved in the replication configuration and that user ID must have the authority to set up replication. Your user ID does not need to be the same on all systems, although it would be easier for you if it was. Setting up replication involves creating objects (such as control tables and table spaces), binding plans (on Linux, UNIX, Windows, and z/OS), creating SQL packages (on OS/400), and running generated SQL to create tables, registrations, and subscription sets. You can use one authorized user ID on all servers in your replication environment, or you can use a different one on each server.

Requirements for Linux, UNIX, Windows, z/OS

Ensure that the user IDs that you use to set up replication can perform the following tasks:

- Connect to all the servers (source server, Capture control server, Apply control server, Monitor control server, target server).
- Select from catalog tables on the source server, Capture control server, Monitor control server, and target server.
- Create tables (including replication control tables), table spaces, and views at the source server, Monitor control server, Capture control server, and Apply control server.
- If you use the DB2 Replication programs to create new target tables: Create tables and table spaces on the target server. (Not required if you use existing tables as targets).
- Bind plans or create packages on each DB2 database involved in replication, including the source server, target server, Monitor control server, and Apply control server.
- Create stored procedures using a shared library and call stored procedures (Linux, UNIX, Windows only).

For non-DB2 relational databases, the user ID must be able to do the following actions:

- Create tables.
- Create Capture triggers on source tables and control tables.
- Create procedures.
- Create nicknames on the DB2 federated database.
- Create sequences (for Oracle databases only).
- Select from catalog tables.

Most replication administrators have DBADM or SYSADM privileges. On DB2 for z/OS the replication administrator should be at least authorized to select from the catalog and should have all privileges necessary to create tables with the ASN schema and to create CD and target tables with the characteristics of the source tables, including index creation privileges.

Requirements for OS/400

Ensure that the user IDs you use to set up replication can perform the following tasks:

- Connect to all the servers (source server, Capture control server, Apply control server, Monitor control server, target server).
- Select from catalog tables on the source server, Capture control server, Monitor control server, and target server.
- Create tables (including replication control tables) and views at the source server, Monitor control server, Capture control server, and Apply control server.
- If you use the DB2 Replication programs to create new target tables: Create tables on the target server. (Not required if you use existing tables as targets.)
- Bind plans or create packages on each DB2 database involved in replication, including the source server, target server, Monitor control server, and Apply control server.

Most replication administrators have DBADM or SYSADM privileges.

Use the Grant DPR Authority (**GRTDPRAUT**) command to authorize a user to register sources, subscribe to those sources, and create control tables. If you are replicating only between OS/400 systems, you should use the same user ID for all servers. See "GRTDPRAUT: Authorizing users (OS/400)" on page 366 for command syntax and parameter descriptions.

If the Grant DPR Authority (**GRTDPRAUT**) command is not installed on a machine, you must use the Grant Object Authority (**GRTOBJAUT**) command.

Authorization requirements for the Capture program

The user ID that runs the Capture program must be able to access the DB2 system catalog, access and update all replication control tables on the Capture control server, and execute the Capture program packages. You can use the replication administrator user ID to run the Capture program, but this is not a requirement.

Requirements for Linux, UNIX, Windows

Ensure that the user IDs that run the Capture program have the following authorities and privileges:

- DBADM or SYSADM authority.
- WRITE privilege on the capture path directory, because the Capture program creates diagnostic files in the **capture_path** directory that you specify when you started the Capture program.

Requirements for z/OS

The user ID used to run the Capture program *must* be registered with access to USS. That means the user ID must be defined to use z/OS UNIX or OS/390 UNIX (it must have an OMVS segment).

Also, ensure that the Capture load library is APF-authorized and that the user ID that runs the Capture program has the following privileges:

- WRITE access to a temporary directory; either the /tmp directory or the directory specified by the TMPDIR environment variable.
- SELECT, UPDATE, INSERT, and DELETE privileges for all replication tables on the Capture control server. (See "List of tables used at the Capture control server" on page 428 for a list of these tables.)
- SELECT privilege for the DB2 catalog (SYSIBM.SYSTABLES and SYSIBM.SYSCOLUMNS).
- TRACE privilege.
- MONITOR1 and MONITOR2 privilege.
- EXECUTE privilege for the Capture program packages.

Also, ensure that the user ID has WRITE access to the capture path directory (USS) or high-level qualifier (z/OS). To run the Capture program in the USS shell, the STEPLIB system variable must be set and it must include the Capture load library. The HFS path, /usr/lpp/db2repl_08_01/bin, must be in your PATH.

Requirements for OS/400

Use the Grant DPR Authority (**GRTDPRAUT**) command to authorize a user to run the Capture program on a local system. See "GRTDPRAUT: Authorizing users (OS/400)" on page 366 for command syntax and parameter descriptions. If you are replicating between only OS/400 systems, you should use the same user ID for all servers. If the **GRTDPRAUT** command is not installed on a machine, you must use the Grant Object Authority (**GRTOBJAUT**) command.

Authorization requirements for Capture triggers on non-DB2 relational databases

If you are replicating from a non-DB2 RDBMS, Capture triggers are used to capture changes from the source. Remote user IDs (for example, from user applications) that change the remote source tables need authority to make inserts into the CCD table. In most cases, you do not need explicit authority to execute INSERT, UPDATE, or DELETE triggers because, after the triggers are defined on a table, the execution of the triggers is transparent to the application that is performing the INSERT, UPDATE, or DELETE. In the case of Informix databases, the remote user IDs that perform INSERT, UPDATE, and DELETE actions against the registered source table need EXECUTE PROCEDURE privilege.

Authorization requirements for the Apply program

The user ID that runs the Apply program must be able to access the DB2 system catalog, access and update all replication control tables on the Capture control and target server, and execute the Apply program packages. You can use the replication administrator user ID to run the Apply program, but this is not a requirement.

Requirements for Linux, UNIX, Windows

Ensure that the user IDs that run the Apply program have the following authorities and privileges:

- WRITE privileges to the apply path directory
- Access privileges to the replication source tables (including associated CD and CCD tables).
- Access and update privileges to the replication target tables.
- Access and update privileges to all control tables that are generated by DB2 replication programs and built at the Capture control server and the Apply control server.
- READ privileges for any password file used by the Apply program.

Note: If your source tables are on a non-DB2 relational database management system: The user ID must have sufficient privileges in both the DB2 federated database *and* in the non-DB2 relational database to access the source tables through nicknames, which are defined on the federated database.

Requirements for z/OS

Ensure that the user IDs that run the Apply program have the following authorities and privileges:

- WRITE access to a temporary directory; either the /tmp directory or the directory specified by the TMPDIR environment variable.
- SELECT, UPDATE, INSERT, and DELETE privileges for all replication tables on the Apply control server. (See "List of tables used at the Apply control server" on page 431 for a list of these tables.)
- SELECT authority for the DB2 catalog (SYSIBM.SYSTABLES and SYSIBM.SYSCOLUMNS).

Note: The user ID used to run the Apply program *must* be registered with access to USS. That means the user ID must be defined to use z/OS UNIX or OS/390 UNIX (it must have an OMVS segment). The load library must be APF-authorized *only* if the Apply program is to be registered with ARM. To run the Apply program in the USS shell, the STEPLIB system variable must be set and it must include the apply load library. The HFS path, /usr/lpp/db2repl_08_01/bin, must be in your PATH.

Requirements for non-DB2 relational database management systems

If your control tables are on non-DB2 relational database management systems, the user ID that is pushing changed data to a non-DB2 relational target or pulling data from it must have sufficient privileges in the DB2 federated database *and* in the non-DB2 relational database.

For non-DB2 relational targets, the user ID running the Apply program needs the privilege to WRITE to nicknames on the DB2 federated database and, through user mappings, the privilege to WRITE to the actual non-DB2 target.

For non-DB2 relational sources, the ID running the Apply program needs the following privileges:

- Privilege to READ from and WRITE to nicknames on the DB2 federated database and, through user mappings, the privilege to READ from and WRITE to the Capture control tables.

- Privilege to READ from nicknames on the DB2 federated database and, through user mappings, the privilege to READ from the actual CCD table on the non-DB2 server.
- Privilege to READ from nicknames on the DB2 federated database and, through user mappings, the privilege to READ from the actual source table on the non-DB2 server.

Requirements for OS/400

Use the Grant DPR Authority (**GRTDPRAUT**) command to authorize a user to run the Apply program on a local system. If you are replicating only between OS/400 systems, you should use the same user ID for all servers. If the **GRTDPRAUT** command is not installed on a machine, you must use the Grant Object Authority (**GRTOBJAUT**) command. See "GRTDPRAUT: Authorizing users (OS/400)" on page 366 for command syntax and parameter descriptions.

You can use different user IDs at each server in your replication environment.

Authorization requirements for the Replication Alert Monitor

The user ID that runs the Monitor program must be able to access and update all replication control tables on the Monitor control server, and execute the Monitor program packages. You can use the replication administrator user ID to run the Monitor program, but this is not a requirement.

Requirements for Linux, UNIX, Windows

Ensure that the user ID that starts the Replication Alert Monitor is a valid logon ID on the Monitor control server where the Monitor control tables reside, and on the servers that contain the control tables that you are monitoring. Also, ensure that the user ID that runs the Replication Alert Monitor has the following authority and privileges:

- SELECT, UPDATE, INSERT, and DELETE privileges for Monitor control tables on the Monitor control servers. (See "List of control tables at the Monitor control server" on page 432 for a list of these tables.)
- SELECT authority on the Capture and Apply control tables that reside on the servers that you want to monitor.
- BINDADD authority (required only if you want to use the autobind feature for the monitor packages).
- EXECUTE privilege for the Monitor program packages.
- WRITE privilege on the monitor path directory where the Replication Alert Monitor stores diagnostic files.
- READ access to the password file used by the Replication Alert Monitor.

Requirements for z/OS

Ensure that the user IDs that run the Monitor program have the following authorities and privileges:

- WRITE access to a temporary directory; either the /tmp directory or the directory specified by the TMPDIR environment variable.
- SELECT, UPDATE, INSERT, and DELETE privileges for all replication tables on the Monitor control server.
- SELECT authority for the DB2 catalog (SYSIBM.SYSTABLES and SYSIBM.SYSCOLUMNS).

Note: The user ID used to run the Monitor program *must* be registered with access to USS. That means the user ID must be defined to use z/OS

UNIX or OS/390 UNIX (it must have an OMVS segment). The load library must be APF-authorized *only* if the Monitor program is to be registered with ARM. To run the Monitor program in the USS shell, the STEPLIB system variable must be set and it must include the monitor load library. The HFS path, /usr/lpp/db2repl_08_01/bin, must be in your PATH.

Storing user IDs and passwords for replication (Linux, UNIX, Windows)

If your replication environment is not distributed across servers, you don't need to store user IDs and passwords. In most replication environments; however, data is distributed across servers. If you have such an environment, when you try to connect to a database, you must provide a valid user ID and password so that DB2 can verify your identity. You store the password information differently for the Replication Center and the other replication programs

You use the **asnpwd** command to create and maintain a password file so that the Apply program, the Replication Alert Monitor, and the Replication Analyzer can access data on remote servers. (The Capture program does not require a password file.) The information in the password file is encrypted to ensure confidentiality. See "asnpwd: Creating and maintaining password files" on page 299 for command syntax and parameter descriptions.

For information about password requirements for the Replication Center, see the Replication Center help and "Managing user IDs and passwords for the Replication Center" on page 223.

Setting up the replication control tables

You can create control tables that are used for replication.

- "Creating control tables (Linux, UNIX, Windows)"
- "Creating control tables (z/OS)"
- "Creating control tables (OS/400)" on page 23
- "Creating control tables for non-DB2 relational sources" on page 23
- "Creating multiple sets of Capture control tables" on page 24
- "Capture control tables on multiple database partitions" on page 24

Creating control tables (Linux, UNIX, Windows)

Use the Replication Center to create replication control tables for the Capture and Apply programs on Linux, UNIX and Windows. When you create the replication control tables, if you do not customize the way that the control tables are created, two table spaces are created, one for the UOW table and one for the other control tables. If you do not want to use the default replication table spaces, you can specify existing table spaces, create new table spaces, or use the current DB2 default table space. See the Replication Center online help for details about creating replication control tables.

If Capture is started in a multiple database partition environment, Capture creates an additional control table (IBMSNAP_PARTITIONINFO) in the same table space as the IBMSNAP_RESTART table.

Creating control tables (z/OS)

Use the Replication Center to create replication control tables on z/OS. You can create a profile for z/OS operating systems to identify the defaults to be used

when you create control tables for that type of system. After you set the profiles for these control tables, you do not have to set them for every set of control tables that you create; however, you can override the defaults when you create the control tables. You can also modify the profile at any time, but the changes will affect only the control tables that you create *after* you modified the profile. See the Replication Center online help for details about creating replication control tables.

Creating control tables (OS/400)

Replication control tables are created automatically when you install DB2 DataPropagator for iSeries. These tables are created in the DataPropagator default schema (called ASN), if they do not already exist.

You can create a new set of Capture control tables with a new Capture schema. You can create a maximum of 25 schemas. Use the Create DPR Tables (**CRTDPRTBL**) command, as described in "Creating multiple sets of Capture control tables" on page 24. You can also use the **CRTDPRTBL** command if your replication control tables are accidentally deleted or corrupted. For details about this command, see"CRTDPRTBL: Creating the replication control tables (OS/400)" on page 360.

Important: Use only the **CRTDPRTBL** command to create control tables on OS/400. The Replication Center does not support the creation of control tables for OS/400.

For a user-defined file system, you can create the replication control tables in the base Auxiliary Storage Pool (ASP) or in Independent Auxiliary Storage Pool (IASP) groups, but not in both. If you create control tables in an IASP group, you must first remove all Capture and Apply control tables from the base ASP. Issue the **SETASPGRP** command for the ASP group that contains the ASN library (or any other library for a Capture schema) before you start the Capture or Apply programs.

Creating control tables for non-DB2 relational sources

If you want to replicate *from* a non-DB2 RDBMS, such as Informix, you must use the Replication Center to create control tables, just as you would if you are replicating from DB2. For these types of sources, the Replication Center creates the following Capture control tables in the non-DB2 relational database:

- Prune control table (IBMSNAP_PRUNCNTL)
- Prune set table (IBMSNAP_PRUNE_SET)
- Register synchronization table (IBMSNAP_REG_SYNCH)
- Register table (IBMSNAP_REGISTER)
- Sequencing table (IBMSNAP_SEQTABLE), on Informix only
- Signal table (IBMSNAP_SIGNAL)

Nicknames are created in a federated database for all but the sequencing table (IBMSNAP_SEQTABLE). (The sequencing table is used only by the Informix triggers. The Apply program doesn't use it.) Triggers are created automatically on the signal table (IBMSNAP_SIGNAL) and the register synchronization table (IBMSNAP_REG_SYNCH).

Important: Do not remove or modify the triggers that are created on the IBMSNAP_SIGNAL and IBMSNAP_REG_SYNCH tables.

Creating multiple sets of Capture control tables

If you want to use more than one Capture program on a server you must create more than one set of Capture control tables and ensure that each set of tables has a unique Capture schema. This schema identifies the Capture program that uses a set of tables. Multiple Capture schemas enable you to run multiple Capture programs concurrently.

You might want to run multiple Capture programs in the following situations:

- To optimize performance by treating low-latency tables differently from other tables. If you have low latency tables, you might want to replicate those tables with their own Capture program. That way, you can give them a different run-time priority. Also, you can set the Capture program parameters, such as pruning interval and monitor interval, to suit the low latency of these tables.
- To potentially provide higher Capture throughput. This can be a significant benefit in a source environment with multiple CPUs. The trade-off for the higher throughput is additional CPU overhead associated with multiple log readers.

If you want to replicate from multiple non-DB2 source databases within the same federated database, you must create multiple sets of Capture control tables, with each set having its own schema. Or, if you prefer, you can use separate federated databases, in which case the Capture control tables on each server can use the default ASN schema.

On z/OS systems, you can use multiple Capture schemas it you want to work with UNICODE and EBCDIC encoding schemes separately or if you want to run more than one instance of the Capture program on a subsystem. See "Creating control tables (z/OS)" on page 22 for information about creating control tables.

On OS/400 systems, use the Create DPR Tables (**CRTDPRTBL**) command to create the extra set of Capture control tables by using the **CAPCTLLIB** parameter to specify the schema name. For details about this command, see "CRTDPRTBL: Creating the replication control tables (OS/400)" on page 360.

Capture control tables on multiple database partitions

When you create Capture control tables in a multiple partitioned database, all of the table spaces used by those control tables must be on the catalog node. If you use an existing table space, the table space must be non-partitioned and on the catalog node.

If you are starting the Capture program for the first time and select the WARMSI start mode, the IBMSNAP_PARTITIONINFO table does not exist. The Capture program creates this table and a unique index for it in the table space that the IBMSNAP_RESTART table is located. After the IBMSNAP_PARTITIONINFO table is created, the Capture program inserts a row into it for every database partition.

If this is not the first time that you started the Capture program and you select one of the warm start modes, the IBMSNAP_PARTITIONINFO table already exists. If you selected the **One or more partitions have been added since Capture was last run** check box, the Capture program inserts a row into the IBMSNAP_PARTITIONINFO table for every database partition that you added since the Capture program last ran. For information about how to create Capture control tables for multiple database partitions from the Replication Center, see the Replication Center help.

Setting up the replication programs

The following sections explain the steps involved in setting up the replication programs for the servers in your environment:

- "Setting up the replication programs (Linux, UNIX, Windows)"
- "Setting up the Capture and Apply programs (OS/400)" on page 28
- "Setting up the replication programs (z/OS)" on page 30
- "Capture for multiple database partitions" on page 30

Setting up the replication programs (Linux, UNIX, Windows)

Read the following instructions to set up the replication programs:

- "Setting environment variables for the replication programs (Linux, UNIX, Windows)"
- "Preparing the DB2 database to run the Capture program (Linux, UNIX, Windows)" on page 26
- "Optional: Binding the Capture program packages (Linux, UNIX, Windows)" on page 26
- "Optional: Binding the Apply program packages (Linux, UNIX, Windows)" on page 26
- "Optional: Binding the Replication Alert Monitor program packages (Linux, UNIX, Windows)" on page 27

Setting environment variables for the replication programs (Linux, UNIX, Windows)

You must set environment variables before you start and stop the Capture program, the Apply program, or the Replication Alert Monitor program, and before you use the Replication Center or replication system commands.

Procedure:

To set the environment variables:

1. Set the environment variable for the DB2 instance name (DB2INSTANCE) as shown:

   For Windows:

   ```
   SET DB2INSTANCE=db2_instance_name
   ```

   For Linux and UNIX:

   ```
   export DB2INSTANCE=db2_instance_name
   ```

2. If you created the source database with a code page other than the default code page value, set the DB2CODEPAGE environment variable to that code page. See "Configuring national language support (NLS) for replication" on page 12.[2]
3. Optional: Set environment variable DB2DBDFT to the source server.
4. **For Linux and UNIX:** Make sure the library path and executable path system variables specific to your system include the directory where the replication libraries and executables are installed.

2. Capture must be run in the same code page as the database for which it is capturing data. DB2 derives the Capture code page from the active environment where Capture is running. If DB2CODEPAGE is not set, DB2 derives the code page value from the operating system. The value derived from the operating system is correct for Capture if you used the default code page when creating the database.

Preparing the DB2 database to run the Capture program (Linux, UNIX, Windows)

Procedure:

To prepare the DB2 database to run the Capture program:

1. Connect to the Capture control server database by entering:

   ```
   db2 connect to database
   ```

   where *database* is the Capture control server database.
2. Prepare the Capture control server database for roll-forward recovery by issuing the **update database configuration** command (logretain recovery) and the **backup database** command. You might need to increase configuration values based on your installation requirements. For transactions with a large number of rows or very large rows it is recommended to increase the CAPPARMS memory limit parameter.

   For multiple database partition environments, every partition must be set up to allow roll-forward recovery for every node that the Capture control server database is on.

   The following database configuration values are adequate for many large workstation scenarios: APPLHEAPSZ 1000, LOGFILSIZ 4000, LOGPRIMARY 8, LOGSECOND 40, DBHEAP 1000, LOGBUFSZ 16, MAXAPPLS 200.

Optional: Binding the Capture program packages (Linux, UNIX, Windows)

The following steps are optional because the Capture program is bound automatically on Linux, UNIX, and Windows during execution.

Procedure:

To bind the Capture program packages:

1. Connect to the Capture control server database by entering:

   ```
   db2 connect to database
   ```

   where *database* is the Capture control server database.
2. Change to the directory where the Capture program bind files are located.

   Windows:
   drive:\sqllib\bnd

   Linux and UNIX:
   db2homedir/sqllib/bnd

   where *db2homedir* is the DB2 instance home directory.
3. Create and bind the Capture program package to the source server database by entering the following command:

   ```
   db2 bind @capture.lst isolation ur blocking all
   ```

   where `ur` specifies the list in uncommitted read format for greater performance.

These commands create packages, the names of which are in the file capture.lst.

Optional: Binding the Apply program packages (Linux, UNIX, Windows)

On Linux, UNIX and Windows the Apply program is bound automatically during execution. Therefore, the following steps are optional on those operating systems.

Procedure:

To bind the Apply program packages:

1. Change to the directory where the Apply program bind files are located.

   Windows:
   drive:\sqllib\bnd

   Linux and UNIX:
   db2homedir/sqllib/bnd

   where *db2homedir* is the DB2 instance home directory.
2. For each source server, target server, Capture control server, and Apply control server to which the Apply program connects, do the following steps:
   a. Connect to the database by entering:
      ```
      db2 connect to database
      ```
      where *database* is the source server, target server, Capture control server, or Apply control server. If the database is cataloged as a remote database, you might need to specify a user ID and password on the **db2 connect to** command. For example:
      ```
      db2 connect to database user userid using password
      ```
   b. Create and bind the Apply program package to the database by entering the following commands:
      ```
      db2 bind @applycs.lst isolation cs blocking all grant public
      db2 bind @applyur.lst isolation ur blocking all grant public
      ```
      where `cs` specifies the list in cursor stability format, and `ur` specifies the list in uncommitted read format.

   These commands create packages, the names of which are in the files applycs.lst and applyur.lst.

Optional: Binding the Replication Alert Monitor program packages (Linux, UNIX, Windows)

The following steps for binding the packages are optional. The Replication Alert Monitor packages are bound automatically during execution. If you want to specify options or check that all bind processes completed successfully, complete the following steps:

Procedure:

To bind the Replication Alert Monitor program packages:

1. Change to the directory where the Replication Alert Monitor program bind files are located.

   Windows:
   drive:\sqllib\bnd

   Linux and UNIX:
   db2homedir/sqllib/bnd

   where *db2homedir* is the DB2 instance home directory.
2. For each Monitor control server, do the following steps:
   a. Connect to the Monitor control server database by entering:
      ```
      db2 connect to database
      ```

where *database* is the Monitor control server. If the database is cataloged as a remote database, you might need to specify a user ID and password on the **db2 connect to** command. For example:

```
db2 connect to database user userid using password
```

b. Create and bind the Replication Alert Monitor program package to the database by entering the following commands:

```
db2 bind @asnmoncs.lst isolation cs blocking all grant public
db2 bind @asnmonur.lst isolation ur blocking all grant public
```

where `cs` specifies the list in cursor stability format, and `ur` specifies the list in uncommitted read format.

These commands create packages, the names of which are in the files asnmoncs.lst and asnmonur.lst.

3. For each server that you are monitoring and to which the Replication Alert Monitor program connects, do the following steps:

a. Connect to the database by entering:

```
db2 connect to database
```

where *database* is the monitored server. If the database is cataloged as a remote database, you might need to specify a user ID and password on the **db2 connect to** command. For example:

```
db2 connect to database user userid using password
```

b. Create and bind the Replication Alert Monitor program package to the database by entering the following command:

```
db2 bind @asnmonit.lst isolation ur blocking all grant public
```

where `ur` specifies the list in uncommitted read format.

These commands create packages, the names of which are in the file asnmonit.lst.

Setting up the Capture and Apply programs (OS/400)

You must set up your environment if you want to use the Apply program with remote systems on other, non-OS/400 operating systems. The following sections explain the steps involved in setting up your replication environment:

- "Creating SQL packages to use with remote systems (OS/400)"
- "Granting privileges to the SQL packages" on page 29

Creating SQL packages to use with remote systems (OS/400)

You need to create packages using the **CRTSQLPKG** command in the following cases:

- When using remote journaling. Run the **CRTSQLPKG** command on the system where the Capture program is running, point to the system where the source table is located.
- Before using the **ADDDPRSUB** or **ADDDPRSUBM** command to add a subscription set or subscription set member. Run the **CRTSQLPKG** command on the target server:
  - If the source table is on a different machine, point to the system where the source table is located.
  - If the Apply control server is on a different machine, point to the Apply control server.

The SQL packages allow replication programs to operate in a distributed replication environment, whether that environment is one in which you are replicating between OS/400 systems or between an OS/400 system and some other operating system (such as Linux, UNIX or Windows).

For information about using the **CRTSQLPKG** command, see *DB2 Universal Database for iSeries SQL Programming*.

The packages are created using the ASN qualifier. On OS/400 they are created in the ASN library. On other operating systems, they are created in the ASN schema.

Creating SQL packages for the Apply program: You must create SQL packages so that the Apply program can interact with all the remote servers to which it needs to connect. For example, run this command on the system where Apply is running to enable it to connect to a remote system:

```
CRTSQLPKG PGM(QDP4/QZSNAPV2) RDB(remote_system)
```

where *remote_system* is the relational database entry name for the remote system to which the Apply program needs to connect.

Creating SQL packages for the Replication Analyzer: You must create SQL packages so that the Replication Analyzer can interact with the servers that you are analyzing, such as the Capture control server or the target server. Run this command on the system where the Replication Analyzer is running:

```
CRTSQLPKG PGM(QDP4/QZSNANZR) RDB(remote_system)
```

where *remote_system* is the name of the system that you are analyzing.

Creating SQL packages for the replication administration commands: For replication between OS/400 systems, if you use a remote journal, you must use this command to create packages for the Capture program and for the replication administration commands. Run this command on the system where Capture is running:

```
CRTSQLPKG PGM(QDP4/QZSNSQLF) RDB(source_system) OBJTYPE(*SRVPGM)
```

where *source_system* is the name of the system where the source table actually exists.

Granting privileges to the SQL packages

After you create the packages, you must grant *EXECUTE privileges to all users who will be subscribing to files registered on the source database. Log on to the OS/400 system where the source database resides and use one of the following methods:

- Use the Grant Object Authority (**GRTOBJAUT**) command:

  ```
  GRTOBJAUT OBJ(ASN/package_name) OBJTYPE(*SQLPKG)
            USER(subscriber_name) AUT(*OBJOPR *EXECUTE)
  ```

- Use SQL to connect to the source database and run the GRANT SQL statement:

  ```
  CONNECT TO data_server_RDB_name
  GRANT EXECUTE ON PACKAGE ASN/package_name TO subscriber_name
  ```

- Use the **GRTDPRAUT** command, if it is installed on the local system. See "GRTDPRAUT: Authorizing users (OS/400)" on page 366 for command syntax and parameters.

Setting up the replication programs (z/OS)

You must set up and customize the replication programs when you install IBM DB2 DataPropagator for z/OS. See the instructions in *Program Directory for IBM DB2 DataPropagator for z/OS*.

Capture for multiple database partitions

If you are replicating data on the DB2 Enterprise Server Edition, you can capture changes to source tables that are spread across multiple database partitions. The Capture program keeps a list of database partitions belonging to its partition group in the IBMSNAP_PARTITIONINFO table. This table is created by the Capture program when the Capture program is started for the first time and finds that there is more than one database partition in its partition group.

Whenever the Capture program is warm started, Capture reads the list of database partitions for the partition group in which its control tables are located. Capture compares the number of database partitions known to DB2 with the number of database partitions listed in the IBMSNAP_PARTITIONINFO table. The number of database partitions listed in the IBMSNAP_PARTITIONINFO table must match the number known to DB2; otherwise, the Capture program will not run.

If you have added one or more database partitions since the last time you ran the Capture program, you must tell the Capture program about the new database partitions. You can do this in the Replication Center by selecting the **One or more partitions have been added since Capture was last run** check box when you set the STARTMODE option to any of the warm start modes on the Start Capture window. For information on how to set up Capture for multiple database partitions from the Replication Center, see the Replication Center help.

Setting up journals (OS/400)

DB2 DataPropagator for iSeries uses the information that it receives from the journals about changes to the data to populate the CD and UOW tables for replication.

DB2 DataPropagator for iSeries runs under commitment control for most operations and therefore requires journaling on the control tables. (The QSQJRN journal is created when the **CRTDPRTBL** command creates a collection.)

Administrators must make sure the libraries containing the source table, CD table, and target table contain journals. They must also ensure that all the source tables are journaled correctly.

Before you register a table for replication on OS/400, the table must be journaled for both before-images and after-images.

The following sections describe the journal setup required for replication:

- "Creating journals for source tables (OS/400)"
- "Managing journals and journal receivers (OS/400)" on page 32

Creating journals for source tables (OS/400)

To set up the source table journals, you must have the authority to create journals and journal receivers for the source tables to be defined. (Skip this section if your source tables are already journaled.)

Important: Use a different journal for the source tables than one of those created by DB2 DataPropagator for iSeries in the ASN (or other capture schema) library.

Procedure:

To create a source table journal:

1. Create a journal receiver in a library of your choice using the Create Journal Receiver (**CRTJRNRCV**) command. Place the journal receiver in a library that is saved regularly. Choose a journal receiver name that can be used to create a naming convention for future journal receivers, such as RCV0001. You can use the *GEN option to continue the naming convention when you change journal receivers. This type of naming convention is also useful if you choose to let the system manage the changing of your journal receivers. The following example uses a library named JRNLIB for journal receivers.

   ```
   CRTJRNRCV  JRNRCV(JRNLIB/RCV0001)
              THRESHOLD(100000)
              TEXT('DataPropagator Journal Receiver')
   ```

2. Create the journal by using the Create Journal (CRTJRN) command:

   ```
   CRTJRN  JRN(JRNLIB/DJRN1)
           JRNRCV(JRNLIB/RCV0001)
           MNGRCV(*SYSTEM) DLTRCV(*YES)
           TEXT('DataPropagator Journal')
   ```

   - Specify the name of the journal receiver that you created in step 1.
   - Use the Manage receiver (MNGRCV) parameter to have the system change the journal receiver and attach a new one when the attached receiver becomes too large. If you choose this option, you do not need to use the **CRTJRN** command to detach receivers and create and attach new receivers manually.
   - Use the default attribute MINENTDTA(*NONE). Other values are not valid for this keyword.
   - Specify DLTRCV(*NO) only if you have overriding reasons to do so (for example, if you need to save these journal receivers for recovery reasons). If you specify DLTRCV(*YES), these receivers might be deleted before you have a chance to save them.

   You can use two values on the RCVSIZOPT parameter of the **CRTJRN** command (*RMVINTENT and *MINFIXLEN) to optimize your storage availability and system performance. See the *OS/400 Programming: Performance Tools Guide* for more information.

3. Start journaling the source table using the Start Journal Physical File (**STRJRNPF**) command, as in the following example:

   ```
   STRJRNPF FILE(library/file)
            JRN(JRNLIB/DJRN1)
            OMTJRNE(*OPNCLO)
            IMAGES(*BOTH)
   ```

   Specify the name of the journal that you created in step 2. The Capture program requires a value of *BOTH for the IMAGES parameter.

4. Change the source table journaling setup:
   a. Use IMAGES(*BOTH) to make sure that the source table is journaled for both before- and after-images.
   b. Make sure that the journal has the following attributes: MNGRCV(*SYSTEM) and DLTRCV(*YES).
   c. Make sure that the journal has the MINENTDTA(*NONE) attribute.

d. For journals on remote systems, specify the MNGRCV(*SYSTEM), DLTRCV(*YES), and MINENTDTA(*NONE) attributes on the source journal. To define the remote journal, specify the DLTRCV(*YES) attribute on the **ADDRMTJRN** command.

Managing journals and journal receivers (OS/400)

The Capture program uses the Receive Journal Entry (**RCVJRNE**) command to receive journals. The following sections describe how to manage journals and journal receivers in your replication environment:

- "Specifying system management of journal receivers (OS/400)"
- "Changing definitions of work management objects (OS/400)"
- "Specifying user management of journal receivers"
- "Using the delete journal receiver exit routine" on page 33

Specifying system management of journal receivers (OS/400)

Recommendation: Let the OS/400 system manage the changing of journal receivers. This is called *system change journal management*. Specify MNGRCV(*SYSTEM) when you create the journal, or change the journal to that value. If you use system change journal management support, you must create a journal receiver that specifies the threshold at which you want the system to change journal receivers. The threshold must be at least 5 000 KB, and should be based on the number of transactions on your system. The system automatically detaches the receiver when it reaches the threshold size and creates and attaches a new journal receiver if it can.

Restriction: When you use the **RTVJRNE** command to retrieve journal entries, no more than 299 source physical files can use the same journal and Capture schema. If you need to register more than 299 files in the same journal, break your source registrations into multiple Capture schemas.

Changing definitions of work management objects (OS/400)

When you install DB2 DataPropagator for iSeries, the installation program creates an SQL journal, an SQL journal receiver for this library, and work management objects. Table 2 lists the work management objects that are created.

Table 2. Work management objects

Description	Object type	Name
Subsystem description	*SBSD	QDP4/QZSNDPR
Job queue	*JOBQ	QDP4/QZSNDPR
Job description	*JOBD	QDP4/QZSNDPR

You can alter the default definitions for the three types of work management objects or provide your own definitions. If you create your own subsystem description, you must name the subsystem QZSNDPR and create it in a library other than QDP4. See *iSeries Work Management, SC41–5306* for more information about changing these definitions.

Specifying user management of journal receivers

If you specify MNGRCV(*USER) when you create the journal (meaning you want to manage changing your own journal receivers), a message is sent to the journal's message queue when the journal receiver reaches a storage threshold, if one was specified for the receiver.

Use the **CHGJRN** command to detach the old journal receiver and attach a new one. This command prevents `Entry not journaled` error conditions and limits the amount of storage space that the journal uses. To avoid affecting performance, do this at a time when the system is not at maximum use.

You can switch journal receiver management back to the system by specifying `CHGJRN MNGRCV(*SYSTEM)`.

You should regularly detach the current journal receiver and attach a new one for two reasons:

- Analyzing journal entries is easier if each journal receiver contains the entries for a specific, manageable time period.
- Large journal receivers can affect system performance and take up valuable space on auxiliary storage.

The default message queue for a journal is QSYSOPR. If you have a large volume of messages in the QSYSOPR message queue, you might want to associate a different message queue, such as DPRUSRMSG, with the journal. You can use a message handling program to monitor the DPRUSRMSG message queue. For an explanation of messages that can be sent to the journal message queue, see *OS/400 Backup and Recovery*.

Using the delete journal receiver exit routine

When you install DB2 DataPropagator for iSeries , a *delete journal receiver* exit routine (**DLTJRNRCV**) is registered automatically. This exit routine is called any time a journal receiver is deleted, whether or not it is used for journaling the source tables. This exit routine determines whether or not a journal receiver can be deleted.

To take advantage of the delete journal receiver exit routine and leave journal management to the system, specify DLTRCV(*YES) and MNGRCV(*SYSTEM) on the **CHGJRN** or **CRTJRN** command.

Important: If you remove the registration for the delete journal receiver exit routine, you must change all the journals used for source tables to have the DLTRCV(*NO) attribute.

If the journal that is associated with the receiver is not associated with any of the source tables, this exit routine *approves* the deletion of the receiver.

If the journal receiver is used by one or more source tables, this exit routine makes sure that the receiver being deleted does not contain entries that have not been processed by the Capture program. The exit routine *disapproves* the deletion of the receiver if the Capture program still needs to process entries on that receiver.

If you must delete a journal receiver and the delete journal receiver exit routine does not approve the deletion, specify `DLTJRNRCV DLTOPT(*IGNEXITPGM)` to override the exit routine.

Removing the delete journal receiver exit routine: If you want to handle the deletion of journal receivers manually, you can remove the delete journal receiver exit routine by using the following command:

```
RMVEXITPGM EXITPNT (QIBM_QJO_DLT_JRNRCV)
           FORMAT(DRCV0100)
           PGMNBR(value)
```

Procedure:

To determine the PGMNBR value for the **RMVEXITPGM** command:

1. Issue the **WRKREGINF** command.
2. On the Work with Registration Information window, find the entry for exit point QIBM_QJO_DLT_JRNRCV. Enter 8 in the **Opt** field.
3. On the Work with Exit Programs window, find the entry for Exit Program QZSNDREP in library QDP4. The number that you need is under the Exit Program Number heading.

Registering the delete journal receiver exit routine: Use the **ADDEXITPGM** command if you removed the exit point and want to put it back. You must register the exit routine with this command:

```
ADDEXITPGM EXITPNT(QIBM_QJO_DLT_JRNRCV)
               FORMAT(DRCV0100)
               PGM(QDP4/QZSNDREP)
               PGMNBR(*LOW)
               CRTEXITPNT(*NO)
               PGMDTA(65535 10 QSYS)
```

Chapter 3. Registering tables and views as SQL replication sources

With DB2 replication, you identify the tables and views that you want to use as replication sources by registering them. When you register a particular table or view for replication, you create a source of available data that you can later use with different targets for various purposes. The administration tasks described in this chapter help you set up the control information that defines how data is captured from each source based on your replication goals.

When you register a source, you identify the table or view that you want to use as a replication source, which table columns you want to make available for replication, and the properties for how DB2 replication captures data and changes from the source.

For DB2 replication, you can register the following objects as sources:

- A DB2 table
- A non-DB2 relational table through a nickname
- A subset of the data in a table (DB2 or non-DB2 relational)
- A view over a single table (DB2)
- A view that represents an inner join of two or more tables (DB2)

This chapter contains the following sections:

- "Registering DB2 tables as sources"
- "Registering non-DB2 relational tables as sources" on page 37
- "Registration options for source tables" on page 38
- "How views behave as replication sources" on page 52
- "Registering views of tables as sources" on page 54
- "Maintaining CCD tables as sources (IMS)" on page 55

Registering DB2 tables as sources

This section describes how to register DB2 tables as replication sources. DB2 replication supports the following types of DB2 tables as sources:

For Linux, UNIX, and Windows

- DB2 tables that your application maintains
- Catalog tables (for full-refresh-only replication)
- Automatic summary tables
- External CCD tables

For z/OS

- DB2 tables that your application maintains
- Catalog tables
- External CCD tables

For OS/400

- DB2 tables that your application maintains (locally or remotely journaled)

- External CCD tables

For all DB2 sources except for OS/400, the source table DDL requires the DATA CAPTURE CHANGES option. Do not remove this option from your source.

When you register a table as a source, a CD (change-data) table is created for you. The Capture program that is associated with the registered table reads the log for the source and stores inflight changes that occur for registered columns in memory until the transaction commits or rollsback. For a rollback, the changes are deleted from memory. For a commit, the changes are inserted into the CD table as soon as the Capture program reads the commit log record. Those changes are left in memory until the Capture program commits the changes after each Capture cycle. The Capture program does not start capturing data for a DB2 source table until a CAPSTART signal has been issued, either by you or the Apply program.

Note for non-relational source tables: You can register DB2 tables that contain data from non-relational database management systems, such as IMS. To do this, you need an application, such as IMS DataPropagator or Data Refresher, to populate a CCD table with the data from the non-relational database. The application captures changes to the non-relational segments in the IMS database and populates a CCD table. The CCD table must be complete, but it can be either condensed or non-condensed. Like other CCD sources, there is no Capture program that is associated with a CCD source table because the table already stores changed data from the non-relational source table. IMS DataPropagator and Data Refresher products maintain the values in the register (IBMSNAP_REGISTER) table so that the Apply program can read from this source table correctly. If you do not use one of these products to maintain these types of CCD tables and you maintain the tables yourself, see "Maintaining CCD tables as sources (IMS)" on page 55.

Prerequisites:

Capture control tables must already exist on the Capture control server that will process the table that you want to register as a source. If you need to create Capture control tables, see "Setting up the replication control tables" on page 22.

Restrictions (OS/400):

- Because SQL statements are limited to a length of 32,000 characters, you can register only approximately 2000 columns per table; the exact number of columns depends on the length of the column names.
- For a single Capture schema, do not register more than 300 source tables that use the same journal.
- Source tables, CD tables, and journals for the source tables must all be in the same Auxiliary Storage Pool (ASP) as the Capture control tables that contain the registration information for these source tables.

Procedure:

Use one of the following methods to register a DB2 table:

Replication Center

Use the Register Tables window. See the Replication Center help for details.

Tip: To save time when registering, you can set up a source object profile ahead of time for the Capture control server. When you register a table, the

Replication Center then uses the defaults that you defined in the source object profile instead of the Replication Center defaults. This can save you time when registering because you can overwrite the defaults once instead of having to select each table one at a time and change the default settings manually.

System commands for replication (OS/400)
Use the **ADDDPRREG** system command. See "ADDDPRREG: Adding a DPR registration (OS/400)" on page 319 for the syntax of this command and a description of its parameters.

When you register a DB2 table, you identify which table you want to register by specifying the source server, source table name, and the Capture schema. You can register the same table multiple times using different Capture schemas. You can use the default settings for registration, or you can modify the registration options to meet your replication needs. See "Registration options for source tables" on page 38 for a complete list of registration options, their defaults, and an explanation of when you might want to use or change the defaults.

Registering non-DB2 relational tables as sources

This section describes how to register non-DB2 relational tables as replication sources. DB2 replication accesses non-DB2 relational tables by using nicknames.

When you register a non-DB2 relational table as a source, a CCD (consistent-change data) table is created for you. When a change for a registered non-DB2 relational table occurs, the Capture triggers simulate the Capture program and insert the change in the CCD table. The Capture triggers start capturing changes for a non-DB2 relational source table at the time you register the source.

By default, the CCD owner is derived from the schema name of the source table. If you modify the CCD owner so that it does not match the schema name, make sure that the source table owner is authorized to write to the CCD table. If the source table owner cannot update the CCD table, triggers on the source table will not be able to write changes to the CCD table.

Prerequisite:

Capture control tables must already exist on the Capture control server that will process this source. If you need to create Capture control tables, see "Creating control tables for non-DB2 relational sources" on page 23.

Restrictions:

- If you are using a single federated DB2 database to access multiple non-DB2 relational source servers, you must use a different Capture schema for *each* non-DB2 relational source server on that single federated database; no two can be the same. You can register a non-DB2 relational table under only one Capture schema.
- You cannot register columns in non-DB2 relational tables that have data types of LOB or DATALINK. If you register a table that includes these data types, you must register a column subset. See "Registering a subset of columns (vertical subsetting)" on page 39 for details on how to register only a subset of columns.

Procedure:

To register a non-DB2 relational table:

Replication Center

Use the Register Nicknames window. See the Replication Center help for details.

Tip: To save time when registering, you can set up a source object profile ahead of time for the Capture control server. When you register a table, the Replication Center then uses the defaults that you defined in the source object profile for CCD tables and nicknames for CCD tables instead of the Replication Center defaults. This can save you time when registering because you can overwrite the defaults once instead of having to select each table one at a time and change the default settings manually.

When you register a non-DB2 relational table, you identify which table you want to register by specifying the nickname of the source table. You can use the default settings for registration, or you can modify the registration options to meet your replication needs. See "Registration options for source tables" for a complete list of registration options, their defaults, and an explanation of when you might want to use or change the defaults.

Registration options for source tables

This section discusses the various options that you have when registering a table as a replication source. These options are part of the larger task of registering a table. For information on how to register:

- A DB2 table, see "Registering DB2 tables as sources" on page 35.
- A non-DB2 relational table, see "Registering non-DB2 relational tables as sources" on page 37.

When you create views over tables and register the views as sources, the views' registration options are determined by the registration definition of the underlying tables. For details on which characteristics views inherit from the underlying tables and how views behave according to the underlying registration definition, see "How views behave as replication sources" on page 52.

After you choose which table that you want to register, you can identify which columns you want to make available for replication, and you can define properties that determine how registered data from this source will be handled and stored. You can also specify other registration options, such as how you want the Capture program to store source data in the CD table (or how you want the Capture triggers to store data in the CCD table). This section discusses the following options that you can specify when registering tables as sources:

- "Registering a subset of columns (vertical subsetting)" on page 39
- "Full-refresh copying and change-capture replication" on page 39
- "After-image columns and before-image columns" on page 41
- "Before-image prefix" on page 43
- "Stop the Capture program on error" on page 43
- "How the Capture program stores updates" on page 44
- "Preventing the recapture of changes (update-anywhere replication)" on page 45
- "Setting conflict detection (update-anywhere replication)" on page 49
- "Using relative record numbers (RRN) instead of primary keys (OS/400)" on page 51

Registering a subset of columns (vertical subsetting)

Default: All columns are registered for replication

When you define a source table for replication, you don't need to register all the columns in the table for replication; you can register a subset of the columns in your source table. This vertical subset can be useful if you do not want to make all of the columns in the table available for targets to subscribe to. You might also want to select this option if the target tables for this source do not support all data types that are defined for the source table.

To register a subset of the columns, select *only* those columns that you want to make *available* for replication to a target table. The columns that you do not select will not be available for replication to any target table. Because CD (and CCD) tables must contain sufficient key data for some types of target tables (such as point-in-time), make sure that your subset contains the columns that will act as the key columns (primary key or unique index) for the target.

Tip: Register a subset of the columns in the source table only if you are sure that you will never want to replicate the unregistered columns. If you register a subset of the columns at the source and later want to replicate columns that you didn't register, you must then alter your registrations to add unregistered columns. (For non-DB2 relational sources, you must redefine your registrations altogether to add new columns to a registration.) If you plan to have an internal CCD associated with this source, it can be even more difficult to add columns later because registering new columns adds them to the CD table but not the internal CCD. To avoid these problems, you might want to register *all* columns at the source and use the Apply program instead to subset which columns are replicated to the targets. For details on how to subset at the target instead of the source, see "Source columns that you want applied to the target" on page 80.

Full-refresh copying and change-capture replication

Default: Change-capture replication

You can choose whether you want all the data in the source table to be replicated to the targets during each replication cycle (full-refresh-only replication), or whether you want only the changes that occurred since the last time that the targets were updated (change-capture replication).

Full-refresh only replication

When targets subscribe to a source that is registered for full-refresh only replication, the Apply program deletes all data from the target table, copies the data that is in the registered columns at the source, and populates the targets with the source data during each replication cycle. The Capture program is not involved, and there is no CD table; the Apply program reads data directly from the source table.

Tip for smaller tables: You might want to choose full-refresh only replication if you have a very small source table that does not take much time or resources to copy.

Tip for larger tables: If you have larger tables and want to use full-refresh only replication, you might want to use the ASNLOAD exit routine to load your tables faster. See "Refreshing target tables using the ASNLOAD exit routine" on page 135 for details.

Restriction: If you plan to have a condensed target table that subscribes to this source and you cannot come up with a unique index for that target table, you must register the source for full-refresh only replication.

Change-capture replication

Default: Changes to all rows are captured

During change-capture replication, only changed data is replicated to the target table. Depending on the type of target table you choose for this source, you must perform an initial load of the table. In most cases, the Apply program performs an initial full refresh, and then continues with change-capture replication.

If you choose not to allow full refresh for target tables, you must manually reload the table if the source and target tables need to be resynchronized. After the target is loaded with the initial source data, the Capture program captures changes that occur at the source and stores them in the CD table. In change-capture replication for non-DB2 relational sources, the Capture triggers capture changes at the source and store them in the CCD table. The Apply program reads the changes from the CD or CCD table and applies the changes to the targets that subscribe to the registered source.

When you define a DB2 source table for change-capture replication, you might not want to store all changes that occur at the source in the CD table. You can register a row (horizontal) subset that filters the changes so that fewer are captured in the CD table than actually occur at the source. You can select from the following two row-capture rules to determine which changed rows from the source table the Capture program records in the CD table:

- Changes to all rows are captured.
- Changes are captured only if the change occurred in a registered column. (DB2 only)

By default, changes are captured whenever a row is updated for any column (registered or unregistered) at the source table. If you register only a subset of the columns, the Capture program records the row values for the registered columns in the CD table every time a change occurs to the source table, even if the columns that changed are different from the registered columns. Use this default option if you want to keep a history of all changes to the source table. This is the *only* option available for non-DB2 relational sources, the Capture triggers capture all changed rows at the source, even if the change occurs in an unregistered column.

Example: Assume that you have 100 columns in your table and you register 50 of those columns for replication. By default, any time a change is made to *any* of the 100 columns in your table, the Capture program will write a row to the CD table (or the Capture triggers will write a row to the CCD table).

If you have a DB2 source, you might want the Capture program to capture changes for registered columns only. In this case, the Capture program writes a row to the CD table *only* when changes occur to registered columns.

Suggestion: Choose to capture changes for all rows if you need information for auditing purposes, or if changes in the table almost always occur in registered columns only. Choose to capture changes for only registered columns if changes frequently occur that only affect unregistered columns. Use this option if you don't want to keep a history of all changes to the source table.

After-image columns and before-image columns

Default: After-image columns only

When you are registering a source for change-capture replication, you can choose whether you want the Capture program to capture only the after-image value (the value in the column after a change was made) or both the after-image value and the before-image value (the value that was in the column before the change was made). For Linux, UNIX, Windows, and z/OS, you can select whether to capture before-image values for each column in the table. For OS/400, you can select whether to capture before images for all or none of the columns in the table; you cannot select this option for each individual column. The sections below discuss when you should choose each option.

Several non-DB2 relational source tables require you to include only after-image values in the CCD table:

- A Sybase or Microsoft SQL Server table can contain only one column of type TIMESTAMP. When the data source is Sybase or Microsoft SQL Server and the source table has a column of type TIMESTAMP, select after images only for this column when you define it as part of the replication source.

Columns with certain data types do not allow you to include before-image values in the CD table:

- Columns with LOB data types
- Columns with DATALINK data types

Capturing after-image values only

For each column that you register for change-capture replication, you can choose for the Capture program or triggers to record only the after-image value for each change. When you select to capture after-image values only, the CD (or CCD) table contains one column for each changed value, which stores the value of the source column after the change occurred.

You don't need before images if you plan to use only base aggregate and change aggregate target-table types for this source. Before-image columns do not make sense if you plan to use your target table for computed values because there is no before image for computed columns. All other target-table types can make use of before-image columns. See "A computed summary of data or changes at the source" on page 72 for more information on aggregate target tables.

Capturing before-image and after-image values

For each column that you register for change-capture replication, you can choose for the Capture program or triggers to record both the before-image and after-image value for each change. When you select to capture before-image and after-image values, the CD (or CCD) table contains two columns for each changed value: one for the value in source column before the change occurred, and one for the value after the change occurred.

When you choose to store both the before and after images in the CD (or CCD) table, the before-image columns and after-image columns have different values for different actions performed on the source tables:

Action	Column value
Insert	The before-image column contains a NULL value. The after-image column contains the inserted value.

Update The before-image column contains the column value before the change occurred. The after-image value contains the column value after the change occurred.

When you choose to have updates captured as delete and insert pairs, the delete row contains the before image from the update in both the before-image and after-image columns of the row, and the insert row contains NULL values in the before-image column and the after image in the after-image column. See "How the Capture program stores updates" on page 44 for more information on this option.

Delete The before-image and after-image columns contain the column value before the change occurred.

Important for Linux, UNIX, Windows, and OS/400: For columns that have before-images defined, DB2 replication limits column names to 29 characters because the entire column name can have only 30 characters. If the column name is longer, DB2 replication truncates the additional characters from the right by default, unless you have set your profile to truncate from the left. Because DB2 replication adds a before-image column identifier (usually *X*) to target columns and each column name must be unique, you cannot use column names that are longer than 29 characters. For tables that you do not plan to replicate, you can use longer column names, but consider using 29-character names in case you might want to replicate these columns in the future.

Important for z/OS: For tables in DB2 for z/OS, you can use 18-character column names, but DB2 DataPropagator will replace the 18th character with the before-image column identifier in target tables, so you must ensure that the first 17 characters of the name are unique.

The following sections describe cases in which you might want to capture before-image values:

- "For keeping a history of your source data"
- "For update-anywhere configurations with conflict detection"
- "When the key columns at the target are subject to update"

For keeping a history of your source data: If you want to keep data for auditing purposes, you might want to select both before and after images so that you have a record of how the data has changed over a period of time. A set of before-image and after-image copies is useful in some industries that require auditing or application rollback capability.

For update-anywhere configurations with conflict detection: In update-anywhere configurations where conflicts are possible between replica tables (where conflict detection is set to anything other than None), you must register both after-image and before-image columns for the CD table of the replicas so that changes can be rolled back if a conflict occurs.

When the key columns at the target are subject to update: When registering a source, consider the potential target tables that you might define using this table as the source. Typically target tables are condensed and require a column or set of columns that make each row in that target table unique. These unique columns make up what is called the target key. If any of these target key columns might be updated at the source, DB2 replication requires special handling to ensure that the correct rows at the target table are updated. To ensure that DB2 replication updates

the correct rows in the target table with the new key value, you can select to capture both after-images and before-images for the columns that will make up the target key. The Apply program needs the before-image values for these registered columns when it applies the changes of non-key source columns to target key columns in the target table. When applying the changes, the Apply program searches in the target table for the row by looking for the target key values that match the before-image value in the source's CD (or CCD) table, and then it updates that target row with the after-image value in the source's CD (or CCD) table.

Although you register these before-image values when you register the source table or view, DB2 replication does not know that your application will make updates to the target key. Later when you define which targets subscribe to this source (by creating subscription sets), you can specify for the Apply program to perform special updates when applying changes from non-key columns at the source to key columns at the target. See "How the Apply program updates the target key columns with the target-key change option" on page 83 for more information.

Before-image prefix

Default (Replication Center): X

Default (OS/400 system commands): @

If you choose to capture both after-image and before-image columns in the CD (or CCD) table, the after-image column name is the name of the column at the source table, and the before-image column name is the name of the column at the source table with a one-character prefix added to the beginning. You can change the default one-character prefix for the before-image column names. The combination of the before-image prefix and the CD (or CCD) column name must be unambiguous, meaning that a prefixed column name cannot be the same as a current or potential column name in the CD (or CCD) table.

Restriction: You cannot use a blank character as the before-image prefix.

Example: If you use `X` as your before-image prefix and you register a source column named `COL`, you cannot register a column named `XCOL` because it is unclear whether `XCOL` is an actual column name of another source column, or the name of a before-image column with a column name of `COL` and a before-image prefix of `X`.

If you are not replicating any before-image columns for a table, you can choose not to have a before-image prefix and set this property to null.

Stop the Capture program on error

Default: The Capture program stops when it encounters certain errors

When the Capture program detects certain problems when processing registrations, it can either terminate or continue to run. You can choose one of the following options to determine how the Capture program reacts when it encounters certain errors while processing a registered source:

Stop Capture on error
: The Capture program writes an error message in the Capture trace (IBMSNAP_CAPTRACE) table and terminates.

This option stops the Capture program when the following fatal errors occur:

- The CD table space is full.
- SQLCODE-911 error occurs 10 times in a row.
- Unexpected SQL errors occur.

This option does *not* stop the Capture program when certain non-fatal errors occur, for example:

- SQLCODES indicate invalid length of data.
- For Capture programs run under z/OS, the compression dictionary does not exist.

When those non-fatal errors occur, the Capture program invalidates the registrations and keeps running.

Do not stop Capture on error
: The Capture program continues to run when certain errors occur. If it encounters errors during the first time trying to process the source, it does not activate the registration. If the registered source was already active, it stops processing the registration. The registration is stopped in either case. A stopped registration has a value of ″S″ (stopped) in the STATE column of the register (IBMSNAP_REGISTER) control table.

This option does not stop the Capture program when the following non-fatal errors occur:

- The registration is not defined correctly.
- The Capture program did not find the CD table when it tried to insert rows of changed data.
- The DATA CAPTURE CHANGES option on the (non-OS/400) source table was detected as being turned OFF when the Capture program was started or reinitialized.

If the registered state of a subscription-set member is in the stopped state due to an error, the Apply program will not be able to process the set.

How the Capture program stores updates

Default: Updates are stored in a single row in the CD table

You can choose how source updates are stored in the CD (or CCD) table. When the Capture triggers or Capture program captures an update to the source table, it can either save the updated value in a single row in the CD table, or it can store the delete in one row and the insert in another, using two rows in the CD (or CCD) table. By default, the update is stored in a single row. You can use this default to reduce storage and increase performance because only one row is stored in the CD (or CCD) table and is read by the Apply program for each change. However, there are some scenarios where you should instruct the Capture program or triggers to capture updates to the source table as DELETE and INSERT pairs.

You *must* capture updates as DELETE and INSERT statements when your source applications update one or more columns referenced by a predicate on the subscription-set member. Suppose that you plan to define a target that subscribes only to source data with a predicate based on a specific column value (for example, WHERE DEPT = 'J35'). When you change that column (for example, to DEPT='FFK'), the captured change will not be selected for replication to the target because it does not meet the predicate criteria. That is, your new FFK department

will not be replicated because your subscription-set member is based on department J35. Converting the updates to a DELETE and INSERT pair ensures that the target-table row is deleted.

Each captured update is converted to two rows in the CD (or CCD) table for all columns. You might need to adjust the space allocation for the CD (or CCD) table to accommodate this increase in captured data.

Important for DATALINK values: For DATALINK columns defined as ON UNLINK DELETE, the unlink is ignored because a DELETE and INSERT pair is handled within the same transaction. The external file is not deleted, but is updated.

Preventing the recapture of changes (update-anywhere replication)

Default for new source table: Recapture changes

Default for new replica table: Do not recapture changes

Restriction: Tables from non-DB2 relational databases cannot participate in update-anywhere; therefore, this option is for only DB2 sources.

In update-anywhere replication, changes can originate at the master table or at the associated replica tables. When you register a table that you plan to use in update-anywhere replication, DB2 replication assumes that it will be the master table in your configuration. You can use the recapture option to control whether changes that originate from one site and are replicated to a second site are recaptured at that second site and therefore available to be replicated to additional sites. During registration, you set this option for the master table. Later, when you map the master source table with its replica targets, you can set whether changes at the replica are recaptured and forwarded to other tables. (For details about mapping masters to replicas, see "Defining read-write targets (update-anywhere)" on page 77.)

When you are registering the source table that will act as the master in your update-anywhere configuration, you can choose from the following two options:

Recapture changes at master
: Updates to the master that originated at a replica are recaptured at the master and forwarded to other replicas.

Do not recapture changes at master
: Updates to the master that originated at a replica are not recaptured at the master and forwarded to other replicas.

When you are registering the replica table in your update-anywhere configuration, you can choose from the following two options:

Recapture changes at replica
: Updates to the replica that originated at the master are recaptured at the replica and forwarded to other replicas that subscribe to this replica.

Do not recapture changes at replica
: Updates to the replica that originated at the master are not recaptured at the replica and forwarded to other replicas that subscribe to this replica.

Preventing changes from being recaptured can increase performance and reduce storage costs because the Capture program isn't capturing the same changes again for each replica.

The following sections discuss how to decide whether to recapture changes based on your update-anywhere configuration:

- "For masters with only one replica"
- "For multiple replicas that are mutually exclusive partitions of the master"
- "For masters that replicate changes to multiple replicas" on page 47
- "For replicas that replicate changes to other replicas (multi-tier)" on page 47

For masters with only one replica

Master: Do not recapture changes at master

Single replica: Do not recapture changes at replica

If you plan to have only one replica in your update-anywhere configuration, you might want to prevent changes from being recaptured at both the master and replica tables. This setting is optimal if the master table is not a source for other replica tables and the replica is not a source for other replicas (in a multi-tier configuration). If there are only these two tables involved, then a change that originates at the replica does not need to be recaptured at the master, and any change that originates at the master does not need to be recaptured at the single replica.

For multiple replicas that are mutually exclusive partitions of the master

Master: Do not recapture changes at master

Replicas: Do not recapture changes at replicas

If you plan to have several replicas that are partitions of the master table, you might want to prevent changes from being recaptured at both the master and each replica. This setting is optimal if none of the replicas is a source for other replica tables. When replicas are partitions of the master, no two replicas ever subscribe to the same data at the master. Therefore, any change that originates at any replica does not need to be recaptured at the master and forwarded on to the other replicas because only the replica where the change occurred subscribes to that source data.

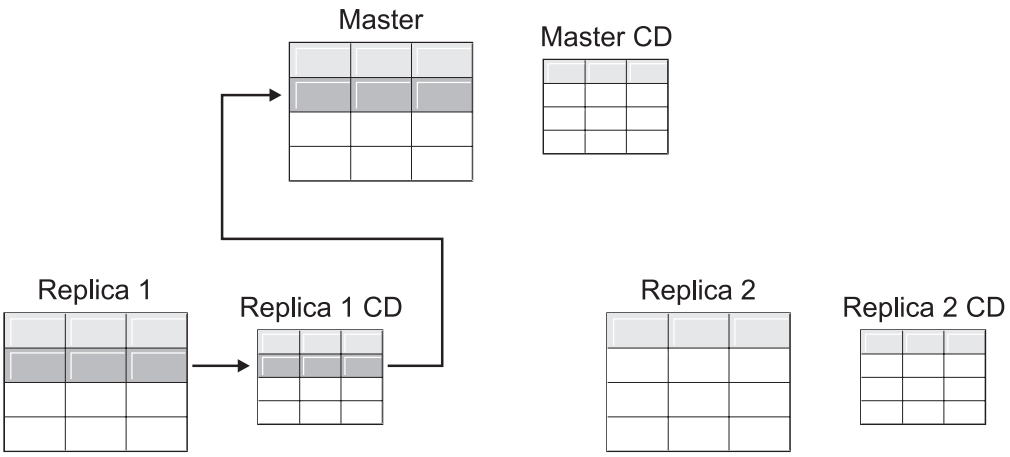


Figure 1. Recapture option for replicas that are mutually exclusive partitions of the master. When you have multiple replicas that do not subscribe to the same data in the master, you do not need to use the recapture option for any of the tables.

For masters that replicate changes to multiple replicas

Master: Recapture changes at master

Replicas: Do not recapture changes at replicas

If you plan to have several replicas that subscribe to the same data in the master table, you might want the Capture program to recapture changes at the master. Changes that originate at a replica are then recaptured at the master and replicated down to other replicas that subscribe to the updated master data.

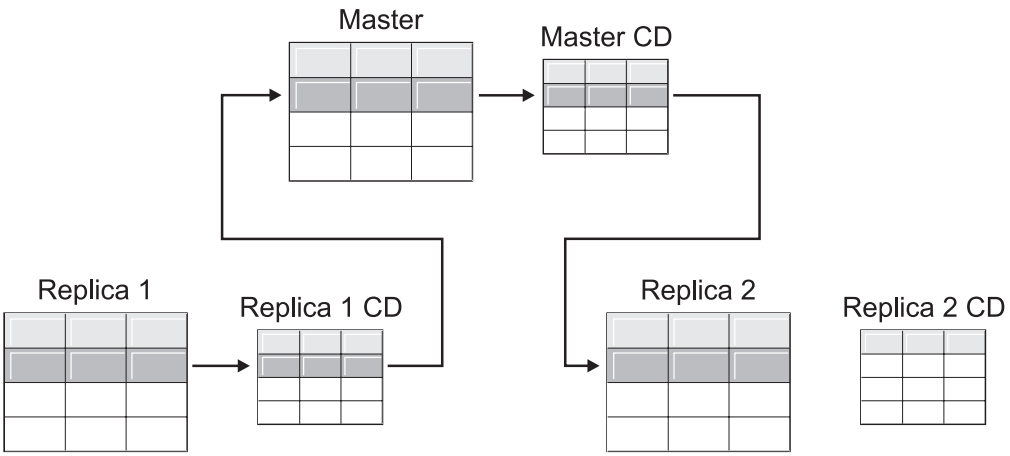


Figure 2. Recapture option for masters that replicate changes to multiple replicas. When you have multiple replicas that subscribe to the same data in the master, you can use the recapture option at the master so that changes that occur at one replica are recaptured at the master and forwarded to the other replica tables.

For replicas that replicate changes to other replicas (multi-tier)

Master: Do not recapture changes at master

Replicas: Recapture changes at replicas

You can have a multi-tier configuration in which the master (tier 1) acts as a source to a replica (tier 2), and then that replica also acts as a source to another replica (tier 3). If you plan to have this type of configuration, you might want the Capture program to recapture changes at the middle replica (tier 2) so that changes that originated at the master are forwarded to the next replica (tier 3).

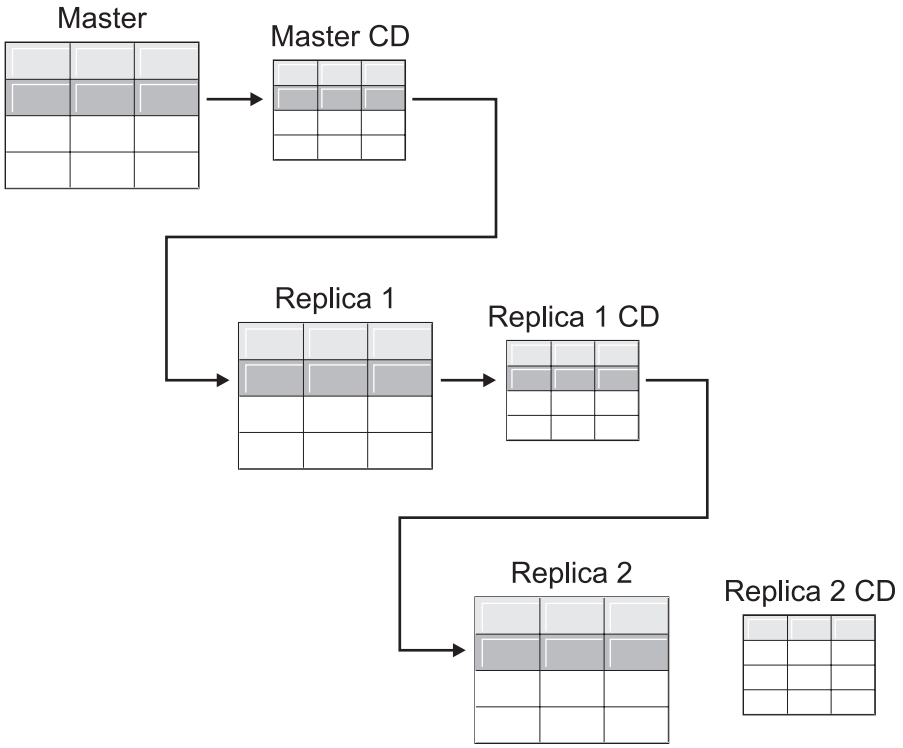


Figure 3. Recapture option at tier 2 allows changes at tier 1 to be replicated down to tier 3. When you have a replica table that acts as a middle tier in a multi-tier configuration, you can use the recapture option at the replica so that changes that occur at the master are recaptured at the replica in the middle tier and forwarded to the replica in the subsequent tier.

Also, when you have recapture set for the middle replica (tier 2), changes that originate at the final replica (tier 3) are recaptured at the middle replica (tier 2) and forwarded to the master (tier 1).

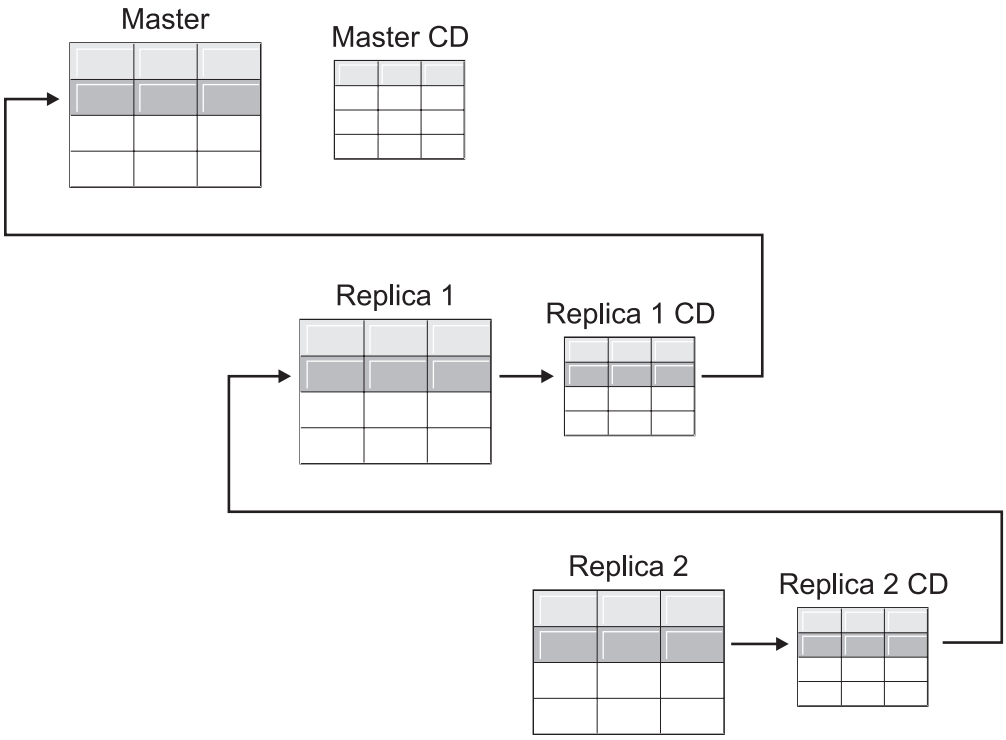


Figure 4. Recapture option at tier 2 allows changes at tier 3 to be replicated up to tier 1. When you have a replica table that acts as a middle tier in a multi-tier configuration, you can use the recapture option at the replica so that changes that occur at the replica in the subsequent tier are recaptured at the replica in the middle tier and forwarded to the master.

Setting conflict detection (update-anywhere replication)

Default: No conflict detection

Restrictions:

- Tables from non-DB2 relational databases cannot participate in update-anywhere; therefore, non-DB2 relational sources do not have conflict detection.
- If you have an update-anywhere configuration that includes DATALINK columns, you must specify None for the conflict-detection level. DB2 does not check update conflicts for external files pointed to by DATALINK columns.
- If you have an update-anywhere configuration that includes LOB columns, you must specify None for the conflict-detection level.

In update-anywhere configurations, conflicts can sometimes occur between the master and its replicas. Conflicts can happen when:

- An update is made to a row in the master table and a different update is made to the same row in one or more replica tables, and the Apply program processes the conflicting changes during the same cycle.
- Constraints are violated.

Although you set the conflict-detection level for individual replication sources, the Apply program uses the highest conflict-detection level of any subscription-set member as the level for all members of the set.

DB2 replication provides three levels of conflict detection: no detection, standard detection, and enhanced detection. Based on your tolerance for lost or rejected transactions and performance requirements, you can decide which type of detection to use. You can select from the following levels of conflict detection when you register a source that you plan to use for update-anywhere replication:

None No conflict detection. Conflicting updates between the master table and the replica table will not be detected. This option is *not* recommended for update-anywhere replication.

Standard
Moderate conflict detection.

During each Apply cycle, the Apply program compares the key values in the master's CD table with those in the replica's CD table. If the same key value exists in both CD tables, it is a conflict. In case of a conflict, the Apply program will undo the transaction that was previously committed at the replica by reading from the replica's CD table and keeping only the changes that originated at the master.

Enhanced
Conflict detection that provides the best data integrity among the master and its replicas.

Like with standard detection, the Apply program compares the key values in the master's CD table with those in the replica's CD table during each Apply cycle. If the same key value exists in both CD tables, it is a conflict. However, with enhanced detection, the Apply program waits for all inflight transactions to commit before checking for conflicts. To ensure that it catches all inflight transactions, the Apply program locks all target tables in the subscription set against further transactions and begins conflict detection after all changes are captured in the CD table. In case of a conflict, the Apply program will undo the transaction that was previously committed at the replica by reading from the replica's CD and keeping only the changes that originated at the master.

Restriction: Even if you specify enhanced conflict detection, when the Apply program runs in an occasionally-connected environment (started with the COPYONCE keyword), the Apply program uses standard conflict detection.

The Apply program cannot detect read dependencies. If, for example, an application reads information that is subsequently removed (by a DELETE statement or by a rolled back transaction), the Apply program cannot detect the dependency.

If you set up a replication configuration where conflicts are possible (by selecting either no detection or standard detection), you should include a method for identifying and handling any conflicts that occur. Even though the replication infrastructure has detected and backed out transaction updates that were in conflict, the application designer must decide what to do about transactions that were at one time committed and now have been backed out. Because the ASNDONE exit routine runs at the end of each subscription cycle, the application designer can use it as a launching point for such application-specific logic. The information regarding conflicting updates that were backed out will remain in the CD and UOW tables until they are eligible for retention limit pruning. For details about how to add this application-specific logic, see "Modifying the ASNDONE

exit routine (Linux, UNIX, Windows, z/OS)" on page 133 or "Modifying the ASNDONE exit routine (OS/400)" on page 134, depending on which platform you are using.

Registering tables that use remote journaling (OS/400)

Default: remote journal is *not* used as the source

When registering OS/400 tables that use remote journaling, you can define for DB2 replication to use the remote journal as the replication source instead of the local journal. By selecting the remote journaling option for replication, you move the CD tables, the Capture program, and the Capture control tables to an OS/400 database server that is separate from the OS/400 server that the source table is on.

When you register tables on OS/400 as sources, the default assumes that you do *not* want to use remote journaling.

Recommendation: Whenever you are replicating data from one OS/400 table to another OS/400 table and you have a remote journal set up, it is highly recommended that you use the remote journaling function when registering. Using remote journaling in replication will greatly increase performance. Because the remote journal function makes it possible to move the registration, the Capture program, and the Capture control tables away from the system on which the source table resides, more resources are left available on that system. This reduces processor usage and saves disk space. Also, when you use a remote journal that resides at the target server, the CD table is on the same system as the target table, which allows the Apply program to apply changes directly from the CD table to the target table without using a spill file. Not using a spill file reduces the amount of resources used by the Apply program.

Recommendation: Register tables that use remote journals as sources only if the registration resides on the same OS/400 system as the replication target. DB2 replication allows you to register remote journals as sources even if the registration does not reside on the same OS/400 system as the target, but then you don't get the performance advantages that you do from having the journal on the target system.

Before you register an OS/400 table that uses remote journaling, make sure that your remote journal is in an active state.

Restriction: Replica target table types are not supported in a remote journal configuration.

For more information about the remote journal function, see *Backup and Recovery*, SC41-5304, and *OS/400 Remote Journal Function for High Availability and Data Replication*, SG24-5189.

Using relative record numbers (RRN) instead of primary keys (OS/400)

Typically, the target table for a source uses the same key columns as the primary key columns in the source. The Apply program uses this key value to track which data it has replicated from the source's CD table to the target. If you are registering an OS/400 table that does not have a primary key, a unique index, or a combination of columns that can be used as a unique index, you must register the table using the relative record numbers (RRN). When you choose to replicate using

the RRN, both the CD table and the target table have an extra column, IBMQSQ_RRN of type INTEGER, which contains a unique value for each row. This column contains the RRN that corresponds to each source table row.

The RRN is used as a primary key for the source table row as long as the source table is not reorganized. When the source table is reorganized, the RRN of each source table row changes; therefore, the RRN in the CD and target table rows no longer has the correct value that reflects the row's new position in the source table. Any time that you reorganize a source table (to compress deleted rows, for example), DB2 DataPropagator for iSeries performs a full refresh of all the target tables in the set of that source table. For this reason, place target tables that use RRN as primary keys in subscription sets with other targets that use RRNs, and not in sets with tables that use some other uniqueness factor.

How views behave as replication sources

When you register views for replication, they inherit the registration options from the registration definitions of their underlying tables. Most importantly, the underlying tables of the view determine whether the view is registered for change-capture replication or full-refresh only. The following sections describe how registered views behave in replication in various scenarios:

- "Views over a single table"
- "Views over a join of two or more tables"

Views over a single table

You can register a view over a single table if the underlying table is registered for replication. When you register a view over a single registered table, the view inherits the type of replication that the underlying table has. If the underlying table is registered for full-refresh-only replication, the view has full-refresh-only replication. You cannot register the view for change-capture replication because the underlying table does not have a CD table associated with it to keep track of changes. If the underlying table is registered for change-capture replication, the view has change-capture replication and cannot be registered for full-refresh only.

When you register a view over a table that is registered for change-capture replication, a view is created for you over the underlying tables' CD table. This CD view contains only the columns referenced by the registered view.

You cannot register a subset of columns in the view; all of the columns in the view are automatically registered.

Views over a join of two or more tables

When you register a view over a join of two or more tables, the underlying tables can be registered or non-registered tables, as long as at least one table in the join is registered. You can also have inner-joins of CCD tables that are registered as sources.

When you register a join as a replication source, DB2 replication adds multiple rows in the register (IBMSNAP_REGISTER) table with identical SOURCE_OWNER and SOURCE_TABLE values. These rows are distinguished by their SOURCE_VIEW_QUAL values. Each of these entries identifies a component of the join.

Restriction: If you define a join that includes a CCD table, all other tables in that join must be CCD tables.

For a join view to be a viable replication source, you must create it using a correlation ID. (Views over single tables do not require a correlation ID.)

Example:

```
create view REGRES1.VW000  (c000,c1001,c2001,c2002,c1003) as
  select a.c000,a.c001,b.c001,b.c002,a.c003
  from REGRES1.SRC001 a, REGRES1.SRC005 b
  where a.c000=b.c000;
```

where VW000 is the name of the view. SRC001 and SRC005 are the tables that are part of the view and C000, C001, C002, and C003 are the columns that are part of the view under the condition that the C000 columns are equal in both tables (SRC001 and SRC005).

The type of replication that the view inherits depends on the combination of its underlying tables, each of which can be:

- Registered for change-capture replication
- Registered for full-refresh-only replication
- Not registered

Table 3 shows the various combinations of underlying tables and what type of source view and CD view results from each combination.

Table 3. Combinations of underlying tables for views

Table 1	Table 2	Description of join view and CD view
Registered for change capture	Registered for change capture	The view is registered for change-capture replication. The CD views contain the referenced columns from Table 1's CD table and from Table 2's CD table.
Registered for change capture	Registered for full-refresh only	The view is registered for change-capture replication. The CD view contains the referenced columns from Table 1's CD table and the referenced columns from Table 2. Only changes to columns that are in Table 1 will be replicated to the registered view's target during each replication cycle.
Registered for full-refresh only	Registered for full-refresh only	The view is registered for full-refresh-only replication. There is no CD view.
Registered for full-refresh only	Not registered	The view is registered for full-refresh-only replication. There is no CD view.
Registered for change capture	Not registered	The view is registered for change-capture replication. The CD view contains referenced columns from Table 1's CD table and the referenced columns from Table 2. Only changes to columns that are in Table 1 will be replicated to the registered view's target during each replication cycle.
Not registered	Not registered	The view is not a valid replication source and cannot be registered.

When you define a view that includes two or more source tables as a replication source, you must take care to avoid double deletes. A double-delete occurs when you delete a row during the same replication cycle from both tables that are part of a view. For example, suppose that you create a view that contains the CUSTOMERS table and the CONTRACTS table. A double-delete occurs if you delete a row from the CUSTOMERS table and also delete the corresponding row

(from the join point of view) from the CONTRACTS table during the same replication cycle. The problem is that, because the row was deleted from the two source tables of the join, the row does not appear in the views (neither base views nor CD-table views), and thus the double-delete cannot be replicated to the target.

To avoid double-deletes, you must define a CCD table for one of the source tables in the join. This CCD table should be condensed and non-complete and should be located on the target server. Defining a condensed and non-complete CCD table for one of the source tables in the join solves the double-delete problem in most situations because the IBMSNAP_OPERATION column in the CCD table allows you to detect the deletes. Simply add an SQL statement to the definition of the subscription set that should run *after* the subscription cycle. This SQL statement removes all the rows from the target table for which the IBMSNAP_OPERATION is equal to "D" in the CCD table.

Problems with updates and deletes can still occur if, during the same Apply cycle, a row is updated on the source table that has the CCD while the corresponding row is deleted on the other table in the join. As a result, the Apply program is unable to find the corresponding row in the joined table and cannot replicate the updated value.

Registering views of tables as sources

This section describes how to register views of DB2 tables as replication sources.

Prerequisites:

- Capture control tables must already exist on the Capture control server that will process the view that you want to register as a source. If you need to create Capture control tables, see "Setting up the replication control tables" on page 22.
- The name of the source views must follow the DB2 table naming conventions.
- You must register at least one of the view's underlying base tables as a source. When you register the base table, use the same Capture schema that you plan to use when you register the view. For information about how to register a table, see "Registering DB2 tables as sources" on page 35.

Restrictions:

- You cannot register views of non-DB2 relational tables.
- You cannot register a view that is over another view.
- On OS/400, because SQL statements are limited to a length of 32,000 characters, you can register only approximately 2000 columns per view; the exact number of columns depends on the length of the column names.
- All CCD tables that have views defined over them must be complete and condensed to be registered as a replication source.

Procedure:

Use one of the following methods to register a view:

Replication Center
Use the Register Views window. See the Replication Center help for details.

System commands for replication (OS/400)
Use the **ADDDPRREG** system command. See "ADDDPRREG: Adding a DPR registration (OS/400)" on page 319 for the syntax of this command and a description of its parameters.

Registration options for views are derived from the registration definition of the source tables on which the views are defined. See "Registration options for source tables" on page 38 for a complete list of registration options, their defaults, and an explanation of when you might want to use or change the defaults. For information on what type of replication that the view will inherit based on its underlying tables (change-capture or full-refresh only), see "How views behave as replication sources" on page 52.

Maintaining CCD tables as sources (IMS)

If you have externally populated CCD tables that are not populated by the Apply program or maintained by a program such as IMS DataPropagator or DataRefresher, you must maintain these tables so that the Apply program can read the CCD tables as sources or function correctly. This section describes how to maintain CCD tables as replication sources.

To maintain a CCD table that is populated by an external tool, you must update three columns in the register (IBMSNAP_REGISTER) table: CCD_OLD_SYNCHPOINT, SYNCHPOINT, and SYNCHTIME. (For details about these columns in the register table, see "*schema*.IBMSNAP_REGISTER" on page 449.) You must update these three columns for each of the following types of events:

- On an initial full refresh or load of the CCD table:
  - Set CCD_OLD_SYNCHPOINT to a value that represents the minimum value of IBMSNAP_COMMITSEQ from the CCD table.
  - Set SYNCHPOINT to a value that represents the maximum value of IBMSNAP_COMMITSEQ from the CCD table. Do not set SYNCHPOINT to 0. If you are creating your own values for sequencing, start with a SYNCHPOINT value of 1.
  - Set SYNCHTIME to a value that represents the maximum timestamp value of IBMSNAP_LOGMARKER from the CCD table.
- On any update to the CCD table after the full refresh or load:
  - Do not change the CCD_OLD_SYNCHPOINT value.
  - Set SYNCHPOINT to a value that represents the new maximum value of IBMSNAP_COMMITSEQ from the CCD table.
  - Set SYNCHTIME to a value that represents the new maximum timestamp value of IBMSNAP_LOGMARKER from the CCD table.
- On any subsequent full refresh or load of the CCD table:
  - Set CCD_OLD_SYNCHPOINT to a value that represents the minimum value of IBMSNAP_COMMITSEQ from the CCD table.
  - Set SYNCHPOINT to a value that represents the maximum value of IBMSNAP_COMMITSEQ from the CCD table.
  - Set SYNCHTIME to a value that represents the maximum timestamp value of IBMSNAP_LOGMARKER from the CCD table.

Important: This assumes that the values that are used in the CCD table for IBMSNAP_COMMITSEQ and IBMSNAP_LOGMARKER are always increasing values. The Apply program will not detect that a full refresh has been performed on the source CCD table unless the CCD_OLD_SYNCHOINT value is larger than the most recently applied SYNCHPOINT value.

Related concepts:

- Chapter 15, "Using the Replication Center for SQL replication," on page 219

Related tasks:

- Chapter 4, "Subscribing to sources for SQL replication," on page 57

Related reference:

- "ADDDPRREG: Adding a DPR registration (OS/400)" on page 319

Chapter 4. Subscribing to sources for SQL replication

After you register the tables and views that you want to use as replication sources, you can define a subscription for your target tables or views so that they receive the source data and the changes from those sources. The administration tasks described in this chapter help you set up the control information that the Capture and Apply programs use to copy source data or to capture changed data and replicate it to the target tables at the appropriate interval.

This chapter consists of the following sections:

- "Planning how to group sources and targets"
- "Creating subscription sets" on page 59
- "Processing options for subscription sets" on page 61
- "Mapping source tables and views to target tables and views within a subscription set" on page 67
- "Selecting a target type" on page 69
- "Common properties for all target table types" on page 80

Planning how to group sources and targets

Before you define which targets subscribe to which sources, you need to plan how you want to group your sources and targets. DB2 replication processes source-to-target mappings in groups. These groups consist of one or more sources that are processed by the same Capture program and one or more targets that subscribe to all or part of the source data, which are processed by the same Apply program. These groups are called *subscription sets*, and the source-to-target mappings are called *subscription-set members*.

When planning for subscription sets, be aware of the following rules and constraints:

- A subscription set maps a source server with a target server. A subscription-set member maps a source table or view with a target table or view. Subscription sets and subscription-set members are stored in the Apply control server.
- The Apply program processes all members in a subscription set as a single group. Because of this, if any member of the subscription set requires full-refresh copying for any reason, all members for the entire set are refreshed.
- All source tables and views in the members of a set must have the same Capture schema.
- On OS/400 systems, all source tables in the members of a subscription set must be journaled to the same journal.
- All external CCD tables created by IMS DataPropagator that are members of a subscription set must have the same Capture schema.

A single Apply program, which has a unique Apply qualifier, can process one or many subscription sets. A single subscription set can contain one or many subscription-set members. The following sections discuss the trade-offs for having few or many sets per Apply program, and few or many subscription-set members per subscription set.

Planning the number of subscription-set members

When you add members to a subscription set, you must decide whether to group all of your source-target pairs (subscription-set members) into one subscription set, create separate subscription sets for each pair, or create a small number of subscription sets, each with a number of pairs.

Because the Apply program replicates the members of a subscription set in one (logical) transaction, you should group multiple members into one subscription set in either of the following situations:

- If the source tables are logically related to one another.
- If the target tables have referential integrity constraints.

By grouping multiple members into one subscription set, you can ensure that replication for all members begins at the same time. Also, you reduce the number of database connections needed to process the subscription sets and you reduce the administration overhead for maintaining your replication environment. If the subscription set contains SQL statements or stored procedures, you can use those statements or procedures to process all of the members of the subscription set.

If there are no logical or referential integrity relationship between the tables in a subscription set, you can group them into one subscription set or into several subscription sets. The main reason for limiting the number of subscription sets is to make administration of the replication environment simpler. But by increasing the number of subscription sets, you minimize the affect of replication failures.

If you want to be able to more easily locate any errors that cause the Apply program to fail, add only a small number of members to a subscription set. With fewer members, you will likely find the source of the problem more quickly than if the set contains a large number of members. If one member of a subscription set fails, all of the data that has been applied to other members of the set is rolled back; so that no member can complete the cycle successfully unless all members do. The Apply program rolls back a failed subscription set to its last successful commit point, which could be within the current Apply cycle if you specified the **commit_count** keyword when you started the Apply program.

Planning the number of subscription sets per Apply qualifier

When you define a subscription set, you specify the Apply qualifier for that subscription set. The Apply qualifier associates an instance of the Apply program with one or more subscription sets. Each subscription set is processed by only one Apply program, but each Apply program can process one or more subscription sets during each Apply cycle.

You can run as many instances of the Apply program (each with its own Apply qualifier) as you need, and each Apply program can process as many subscription sets as you need. You have two basic options:

- Associate each Apply qualifier with one subscription set (each Apply program processes exactly one subscription set)

  If speed is important, you can spread your sets among several Apply qualifiers, which allows you to run several instances of the Apply program at the same time. If you decide to have an Apply-program instance process one subscription set, you can use the Apply program's OPT4ONE startup option, which loads the control-table information for the subscription set into memory. If you use this option, the Apply program does not read the control tables for the subscription-set information for every Apply cycle. Therefore, the Apply

program performs better. However, the more Apply-program instances that you run, the more system resources they will use, and the slower their overall performance might be.

- Associate each Apply qualifier with multiple subscription sets (each Apply program processes many subscription sets)

  By using more than one Apply qualifier, you can run more than one instance of the Apply program from a single user ID.

  The Apply program tries to keep all sets for a given Apply qualifier as current as possible. When an Apply cycle starts, the Apply program determines which of the subscription sets contains the least current data and starts processing that set first.

  If speed is not your main goal, you might want to replicate a large number of subscription sets with one Apply qualifier. For example, this could be a very good option if you wait until after business hours before replicating.

  One disadvantage of having one Apply program process multiple subscription sets is that the Apply program processes the subscription sets sequentially; thus, your overall replication latency can increase.

If you have specific requirements for certain subscription sets, you can combine these two options. For example, you could have one Apply program process most of your subscription sets and thus take advantage of using one Apply program to process related subscription sets together, and you can have another Apply program process a single subscription set and thus ensure minimum replication latency for that subscription set. And by using two instances of the Apply program, you increase the overall parallelism for your subscription sets.

Creating subscription sets

Before you replicate data from a registered source, you must create subscription sets, which are collections of subscription-set members (source-to-target mappings) that the Apply program processes as a set. This section discusses the properties that you define for each subscription set. These properties, which apply to every member that you add to the set, define which servers you want to replicate data to and from, including which Capture program (based on the Capture schema for the registered source) and Apply programs you want to use, and when and how you want the Apply program to process the set.

You don't have to add subscription-set members to a subscription set; instead, you can create an empty set, which is a set that doesn't contain any source-to-target mappings. You might want to create an empty set for the following reasons:

- You plan to add members to a set later and don't plan to activate the subscription set until you add members.
- You want the Apply program to process the empty subscription set in order to call an SQL statement or a stored procedure whenever the set is eligible for processing.

Prerequisites:

1. You must create the Apply control tables in the Apply control server for the subscription set.
2. Before you add subscription-set members to subscription sets, you must register the tables or views that you want to use as sources. If you need to register sources for replication, read and follow the instructions in Chapter 3, "Registering tables and views as SQL replication sources," on page 35. You

should also consider how you want to group your sets. If you need to plan for your sets, see "Planning how to group sources and targets" on page 57 for more information.

Procedure:

To create a subscription set, you can use either of the following two methods:

Replication Center
Use the Create subscription sets notebook. See the Replication Center help for details.

System commands for replication (OS/400)
Use the **ADDDPRSUB** system command. See "ADDDPRSUB: Adding a DPR subscription set (OS/400)" on page 327 for the syntax of this command and a description of its parameters.

To create a subscription set, you provide these basic characteristics:

Apply control server alias

The local alias of the server containing the control tables for the Apply program that will process the subscription set. Define the same alias for the Apply control server in every database from which you run the Replication Center or the Apply program so that the Replication Center will populate the Apply control tables correctly and so that every Apply program will connect to the correct server using a standard alias name.

Subscription set name

The name of the subscription set. At the Apply control server that processes this subscription set, the set name must be unique for a given Apply qualifier. The name can be up to 18 characters long.

Apply qualifier

The name of a new or existing Apply qualifier, which identifies which Apply program will process this subscription set. You can use the same Apply qualifier to process multiple subscription sets. Subscription sets that have the same Apply qualifier must be defined in the same Apply control server. If you are creating a new Apply qualifier, see Chapter 17, "Naming rules for SQL replication objects," on page 269 for the rules on how to name an Apply qualifier.

Capture control server alias

The alias of the server containing the control tables for the Capture program that will process the registered sources for the subscription set. Define the same alias for the Capture control server in every database from which you run the Replication Center or the Apply program so that the Replication Center will populate the Capture and Apply control tables correctly and so that every Apply program will connect to the correct server using a standard alias name.

Capture schema

The name of the Capture schema that identifies the set of Capture control tables that define the registered sources for the subscription set. All of the source tables in a subscription set must reside on the same server, and only one Capture program can be capturing the changes for them.

Target server alias

The name of the target server that contains the tables or views to which the Apply program will replicate changes from the source. Define the same alias for the target server in every database from which you run the Replication Center or the Apply program so that the Replication Center will populate the Apply control tables correctly and so that every Apply program will connect to the correct server using a standard alias name.

When you create a subscription set, you can use the default settings for how the Apply program processes the set, or you can modify the subscription properties to meet your replication needs. See "Processing options for subscription sets" for a complete list of processing options for subscription sets, their defaults, and an explanation of when you might want to use or change the defaults.

Processing options for subscription sets

This section discusses the properties that you can define to specify how the Apply program process the subscription set. In addition, this section helps you to decide which settings to select based on your replication needs:

- "Specifying whether the set is active"
- "Specifying how many minutes worth of data the Apply program retrieves"
- "Specifying how the Apply program replicates changes for members in the set" on page 64
- "Defining SQL statements or stored procedures for the subscription set" on page 65
- "Scheduling the replication of a subscription set" on page 65

Specifying whether the set is active

Default: Inactive

You can specify whether you want the Apply program to begin processing the subscription set. When you activate a subscription set, the Apply program initiates a full refresh for that set. You have three activation levels to choose from:

Active The Apply program processes the set during its next cycle. Activate the set if you want the Apply program to process the set the next time it runs; you can still add members to the set later. When you activate the set, it remains active and the Apply program continues to process it until you deactivate it.

Inactive
The Apply program does not process the set. Leave the set inactive if you are not ready for the Apply program to process it.

Active only once
The Apply program processes the set during its next cycle and then deactivates the set. Specify this option if you want the set to run only once. Make sure that you add all the subscription-set members before selecting this option because the Apply program will not process members that you add later, unless you reactivate the subscription set.

Specifying how many minutes worth of data the Apply program retrieves

Default: 20 minutes

You can specify an approximate number of minutes' worth of data for the Apply program to retrieve from the replication source during each Apply cycle. There are several situations for which this specification can be useful:

- When the amount of data to be processed within one subscription-set cycle is large.

  Subscription sets that replicate large blocks of changes in one Apply cycle can cause the spill files or logs (for the target database) to overflow. For example, batch-Apply scenarios can produce a large backlog of enqueued transactions that need to be replicated.
- An extended outage of the network can cause a large block of data to accumulate in the CD tables, which can cause the Apply program's spill file and the target's log to overflow.

The number of minutes that you specify is called the data block. The data-blocking value that you specify is stored in the MAX_SYNCH_MINUTES column of the subscription sets (IBMSNAP_SUBS_SET) table. If the accumulation of data is greater than the size of the data block, then the Apply program converts a single Apply cycle into several mini-cycles. If resources are still not sufficient to handle the blocking factor provided, the Apply program reduces the size of the data block to match available system resources. By retrieving smaller sets of data, the Apply program can lessen both the network load and the temporary space required for the retrieved data.

During each Apply cycle, if a subscription set's MAX_SYNCH_MINUTES value is NULL, or is set to a numeric value less than 1, the Apply program processes all eligible data for that set in a single Apply cycle. If your CD and UOW tables contain large volumes of data, this situation can lead to such problems as the database transaction log becoming full or a spill file overflowing. You can change MAX_SYNCH_MINUTES to a non-NULL value using the following guidelines:

- If the SLEEP_MINUTES column of the ASN.IBMSNAP_SUBS_SET table is set to 5 minutes (or less) for a given subscription set, set MAX_SYNCH_MINUTES to 5 minutes.
- If SLEEP_MINUTES is set to 30 minutes (or more) for a given subscription set, set MAX_SYNCH_MINUTES to 60 minutes.
- For SLEEP_MINUTES between 5 and 30 minutes, set MAX_SYNCH_MINUTES equal to SLEEP_MINUTES.

Monitor your replication environment and adjust the MAX_SYNCH_MINUTES as needed. Ensure that the numeric value for MAX_SYNCH_MINUTES is greater than zero.

Example: If you specify that the Apply program should retrieve at most 10 minutes' worth of data per mini-cycle, the Apply program will retrieve an amount of committed data from the CD table at the source that is within approximately 10 minutes of the last mini-cycle.

In addition to preventing the logs and spill files from overflowing, these mini-cycles have several other benefits. If there is an error during the replication cycle, the Apply program must roll back only the changes that it made during the mini-cycle that failed. If replication fails during a mini-cycle, the Apply program tries to process the subscription set from the last successful mini-cycle, which can save a significant amount of time if a large amount of changed data is available to be processed. Figure 5 on page 63 shows how the changed data is broken down into subsets of changes.

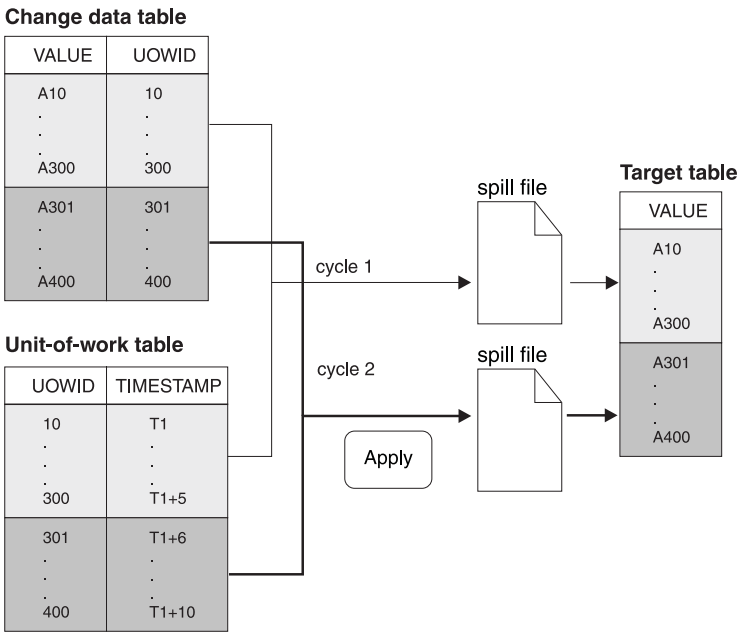


Figure 5. Data blocking. You can reduce the amount of network traffic by specifying a data-blocking value.

The number of minutes that you set should be small enough so that all transactions for the subscription set that occur during the interval can be copied without causing the spill files or log to overflow during the mini-cycle.

When processing data, the Apply program does not take any of the following actions:

- Split a unit of work (meaning that a long running batch job without commits cannot be broken up by the data blocking factor).
- Roll back previously committed mini-subscription cycles.
- Use the data blocking factor during a full refresh.

Deciding how the Apply program loads target tables that have referential integrity

If you want to have referential integrity between target tables in a set, you must choose how the Apply program will process the source data during the initial load of the target tables. By default, the Apply program performs the full refresh of the targets by reading all the rows in the source, storing them in memory, and then inserting the rows into the targets tables. However, you have other options about the way that the target tables are initially loaded. You do not make these decisions when you create subscription sets or define the source-to-target mappings (members) for each set; you decide how the targets will be loaded when you set the startup parameters for the Apply program. You should consider your replication requirements for each subscription-set member while you are defining it and can decide which startup option you will use for the Apply program that will process each member.

Consider one of the following two alternatives on when you can create these referential integrity relationships:

- Before the target tables are populated.

This requires that no changes are made at the source table during the entire extract and load stage of the target table. Also, you must start the Apply program using the LOADX startup option in order to get the speed of the load processing and to bypass the referential constraint checking during this initial population. If you do not use the Apply startup option of LOADX, the inserts in the target table could fail.

- After the Apply program has fully populated the target tables and has processed one complete and successful cycle of applying changes to this set of tables.

  The advantage waiting to add the referential integrity constraints to these tables is that the changes can still be made at the source table while the target tables are being loaded. You can start the Apply program with or without the LOADX startup option, because there are no constraints that need to be bypassed. A full refresh will typically be much faster using the LOADX startup option. During the initial population of the target tables, the targets might be out of synch with each other regarding their referential integrity relationships; but, as they are being loaded, all changes are being captured for the set. After the Apply program replicates the first set of changes, all target tables will contain the same transactions and will have referential integrity. At this point, you can deactivate the set, add the referential integrity constraints, and then reactivate the set.

For more information about the startup options that you have for how your target tables are initially loaded, see "Refreshing target tables using the ASNLOAD exit routine" on page 135.

Specifying how the Apply program replicates changes for members in the set

When the subscription set has change-capture replication, you can decide how you want the Apply program to replicate changes to every source-to-target mapping in the set. After the target tables are initially loaded, the Apply program starts to read the CD (or CCD) tables and collects the changes into spill files. For each CD (or CCD) table, the Apply program creates a separate spill file. The Apply program then reads the changes from the spill files and applies them to the target tables. It can do this in one of three ways:

- Using table-mode processing
- Using transaction-mode processing
- Using a mixture of table-mode and transaction-mode processing, depending on the target-table types in the subscription set

By specifying the type of processing for the subscription set, you can control how often the Apply program commits its changes to the target table or view. The Apply program can commit once for each subscription-set member or after applying a number of transactions. Having one commit can reduce the latency for the subscription set, but having multiple commits allows the Apply program to apply the data in the original commit sequence.

Table mode

The Apply program reads all changes from a spill file for a CD (or CCD) table, applies the changes to the corresponding target tables, and then begins to process the spill file for the next CD (or CCD) table. When it is done reading and applying changes from all the CD (or CCD) tables in the set, it then issues a DB2 commit to commit all of the changes to all of the target tables in the subscription set.

Transaction mode

The Apply program opens all of the spill files at once and processes the changes from them at the same time. The order in which the changes are applied to the target tables is the order in which the transactions took place at the source tables. The COMMIT_COUNT column in the IBMSNAP_SUBS_SET table controls how changes are applied and committed to all target tables for that subscription set. Use the transaction-mode processing when you have referential integrity constraints on the target tables in the subscription set.

You can specify that the Apply program use transaction-mode processing for any subscription set, however, it only changes the Apply program's behavior for sets with user-copy and point-in-time target tables, but not sets with the following types of target tables:

- CCD target tables. Sets containing CCD tables as sources are always processed in table mode.
- Any target table for which the source table is a CCD table. Sets containing CCD tables are always processed in table mode.
- Replica target tables. Sets containing replica tables are always processed in transaction mode.

Defining SQL statements or stored procedures for the subscription set

You can define SQL statements or stored procedures that run each time the Apply program processes the subscription set. These statements can be useful for pruning CCD tables or manipulating source data before it is applied to targets. You can specify when and where the SQL statements or stored procedures should run:

- At the Capture control server before the Apply program applies the data.
- At the target server before the Apply program applies the data.
- At the target server after the Apply program applies the data.

The Apply program processes the statements or procedures in the order listed above.

When you use the Replication Center to add SQL statements to a subscription set, you can click **Prepare statement** in the Add SQL Statement or Procedure Call window to verify the statement's syntax.

Scheduling the replication of a subscription set

You can control how often the Apply program processes a subscription set and thereby control how current the data in your target tables is. You can control how often the subscription set is eligible for processing by using time-based or event-based scheduling, or you can use these scheduling options together. The Apply program begins processing a subscription set when the subscription set is eligible for processing. For example, you can set an interval of one day, and also specify an event that triggers the subscription cycle. If you use both of these scheduling options, the subscription set will be eligible for processing at both the scheduled time and when the event occurs.

In update-anywhere replication, you can use the same or different timing for the master-to-replica and replica-to-master subscription sets.

If there is a large amount of data to be replicated during an interval or between events for any subscription set that the Apply program processes, it is possible for

a particular subscription set to become eligible for processing, but the Apply program will not be able to process it until it finishes applying data for all subscription sets in the prior interval or for the prior event. As soon as it finishes processing the subscription set, the Apply program begins processing the next eligible subscription set. In this case, you might not get the expected replication latency, but you won't lose any data.

Procedure:

To specify the schedule for a subscription set, you can use either of the following two methods:

Replication Center

Use one of the following notebooks:

- Create Subscription Set. Use the **Schedule** page to select the scheduling option that you want.
- Subscription Set Properties. Use this notebook if you have already created the subscription set and want to change the scheduling for the subscription set.

See the Replication Center online help for details.

System commands for replication (OS/400)

Use the **ADDDPRSUB** system command. See "ADDDPRSUB: Adding a DPR subscription set (OS/400)" on page 327 for the syntax of this command and a description of its parameters.

Time-based scheduling

The simplest method of controlling when the set is processed is to use time-based scheduling (also known as relative timing or interval timing). You determine a specific start date, time, and interval. The interval can be specific (from one minute to one year) or continuous, but time intervals are approximate. The Apply program begins processing a subscription set as soon as it is able, based on its workload and the availability of resources. Choosing a timing interval does not guarantee that the frequency of replication will be exactly at that interval. If you specify continuous timing, the Apply program replicates data as frequently as it is able.

Event-based scheduling

To replicate data using event-based scheduling (also known as event timing), you can specify an event name when you define the subscription set. To allow the Apply program to recognize the event when it occurs, you must also populate the subscription events (IBMSNAP_SUBS_EVENT) table with a timestamp for the event name. When the Apply program detects the event, it begins replication.

The subscription events table has four columns, as shown in Table 4.

Table 4. The subscription events table

EVENT_NAME	EVENT_TIME	END_OF_PERIOD	END_SYNCHPOINT
END_OF_DAY	2002-05-01-17.00.00.000000	2002-05-01-15.00.00.000000	

EVENT_NAME is the name of the event that you specify while defining the subscription set. EVENT_TIME is the timestamp for when the Apply program begins to process the set. END_OF_PERIOD is an optional value that indicates that updates that occur after the specified time should be deferred until a future event or time. END_SYNCHPOINT is also an optional value that indicates that updates

that occur after the specified log-sequence number should be deferred until a future event or time. If you specify values for both END_OF_PERIOD and END_SYNCHPOINT, the value for END_SYNCHPOINT takes precedence. Set the EVENT_TIME value using the clock at the Apply control server, and set the END_OF_PERIOD value using the clock at the source server. This distinction is important if the two servers are in different time zones.

In Table 4 on page 66, for the event named END_OF_DAY, the timestamp value for EVENT_TIME (2002-05-01-17.00.00.000000) is the time when the Apply program should begin processing the subscription set. The END_OF_PERIOD timestamp value (2000-05-01-15.00.00.000000) is the time after which updates are not replicated and will be replicated on the next day's cycle. That is, the event replicates all outstanding updates made before three o'clock, and defers all subsequent updates.

You or your applications must post events to the subscription events (IBMSNAP_SUBS_EVENT) table using an SQL INSERT statement to insert a row into the table to activate the event. For example, use the current timestamp plus one minute to trigger the event named by EVENT_NAME. Any subscription set tied to this event becomes eligible to run in one minute. You must manually post events for both full refresh and change-capture replication.

You can post events in advance, such as next week, next year, or every Saturday. If the Apply program is running, it starts at approximately the time that you specify. If the Apply program is stopped at the time that you specify, when it restarts, it checks the subscription events table and begins processing the subscription set for the posted event.

The Apply program does not prune the table; you must populate and maintain this table. Also, you cannot use the Replication Center to update the subscription events table. You must issue SQL statements or define automated procedures to add events to this table.

Example:

```
INSERT INTO ASN.IBMSNAP_SUBS_EVENT
       (EVENT_NAME, EVENT_TIME)
       VALUES ('EVENT01', CURRENT TIMESTAMP + 1 MINUTES)
```

Any event that occurs prior to the most recent time that the Apply program processed the subscription set (as specified by the value in the LASTRUN column of the subscription-set control table) is considered to be an expired event and is ignored. Therefore, if the Apply program is running, you should post events that are slightly in the future to avoid posting an expired event.

Mapping source tables and views to target tables and views within a subscription set

Within a subscription set, you can add source-to-target mappings that the Apply program will process as a group when it processes the set. These source-to-target mappings are called subscription-set members. When defining a subscription-set member, you specify which target table or view subscribes to the source data, and you can define how you want the replicated data to appear at the target.

Prerequisites:

Before you set up targets that subscribe to changes at sources, you must register the tables or views that you want to use as sources. If you didn't already register

sources for replication, read and follow the instructions in Chapter 3, "Registering tables and views as SQL replication sources," on page 35. You should also create a subscription set and plan for how many members you want to add in a set. If you need to create a subscription set, see "Creating subscription sets" on page 59. If you need to plan for subscription-set members, see "Planning the number of subscription-set members" on page 58.

Restrictions:

- DB2 replication does not support views of non-DB2 relational tables as sources.
- If you define a target view, that view must be an insertable view. That is, all of the view's columns must be updateable and the full select for the view cannot include the keywords UNION ALL.
- If you are using the Replication Center, you cannot add a column to a subscription-set member if that column does not already exist in the target table.
- **For Windows, Linux, UNIX, z/OS**: You can define a maximum of 200 members for each subscription set.
- **For OS/400**: You can define a maximum of 78 members for each subscription set.

Procedure:

To add a subscription-set member, you can use either of the following two methods:

Replication Center

Use one of the following notebooks:

- Create Subscription Set. Use this notebook when you create the subscription set.
- Subscription Set Properties. Use this notebook if you have already created the subscription set and want to add one or more subscription-set members to it.
- Add Members to Subscription Sets. Use this notebook to add one member to multiple subscription sets. For example, if you select four subscription sets when you open this notebook, you can add one member to each. Each member must use the same source.

See the Replication Center online help for details.

System commands for replication (OS/400)

Use the **ADDDPRSUBM** system command. See "ADDDPRSUBM: Adding a DPR subscription-set member (OS/400)" on page 343 for the syntax of this command and a description of its parameters.

To map a source with a target, specify the following information about the registered table or view that you want to use as the source:

- The source table or view and a target table or view (including a table space and index for the target table).
- The type of target table.
- The registered columns from the source table that you want to replicate to the target table.

  When you use the Replication Center to map a source with a target, LOB columns and DATALINK columns are not automatically included in the column mapping. You must explicitly select those columns.
- The rows from the source table that you want to replicate to the target table (you include a WHERE clause to specify the rows).

To map the chosen source to a DB2 target, specify the following information about the target table or view:

- The schema of the target table or view.
- The name of the table or view you want to use as the target.

  Default: The default name comes from the target object profile for the target server, if there is one. If you have not set this profile, the default is TG followed by the name of the source table or view. (For example, if the name of your source table is EMPLOYEE, the name of your target table defaults to TGEMPLOYEE.)
- The type of target table

  Default: user copy

If the specified target table does not exist, the Replication Center or the **ADDDPRSUBM** system command creates it.

To map the chosen source to a non-DB2 relational target, specify the following information about the target table:

- The schema of the nickname of the target table
- The nickname of the target table
- The remote schema
- The name of the remote table

  Default: The default name comes from the target object profile for the target server, if there is one. If you have not set this profile, the default is TG followed by the name of the source table or view. (For example, if the name of your source table is EMPLOYEE, the name of your target table defaults to TGEMPLOYEE.)
- The type of target table

  Default: user copy

When you add a subscription-set member, you can use the default target table type of user copy, or you can select another target table type to meet your replication needs.

When you add a subscription-set member for a target table that does not yet exist, you can use the default settings, or you can modify the member properties to meet your replication needs. You can first pick the type of target table that you want to use, and you can then set properties for how the Apply program replicates data to that target. See "Selecting a target type" for a description of various replication scenarios and which target table types you might want to use in each case. The section also helps guide you through which settings to choose based on your replication goals. Regardless of what target type you select, you can modify the common set of properties that all members share. See "Common properties for all target table types" on page 80 for a complete list of options for subscription-set members, their defaults, and an explanation of when you might want to use or change the defaults.

Selecting a target type

This section gives you a description of each of the types of target tables you can choose from, and then it helps you decide which type of target table to select and how to define the target-table properties based on your replication goals. It also discusses what you should do if you want to use an existing table as your target. The type of target table that you need depends on how you want your data to

appear at the target and what replication configuration you have. You can use either an existing table as your target, or you can create a new table.

The names of all non-DB2 relational target tables and indexes must follow the DB2 table and index naming conventions.

Restrictions:

- The null attributes of after-image target columns must be compatible with the null attributes for those columns of the source table or view. Use the SQL COALESCE expression to provide compatibility with existing columns.
- For source tables on non-DB2 relational databases, you can define only the following types of target tables:
  - User copy tables
  - Point-in-time tables
  - External CCD tables
- For source tables on OS/400 systems that use RRN columns as their key columns, you can define only the following types of target tables:
  - Point-in-time tables
  - External CCD tables
- For source tables in a z/OS subsystem, the encoding scheme for the CD and UOW tables must be the same if the Apply program will join these tables to satisfy a subscription-set WHERE clause for a user-copy table. See Appendix A, "UNICODE and ASCII encoding schemes for SQL replication (z/OS)," on page 507 for more information about encoding schemes.

You can select from the following types of target tables:

User copy
: Read-only target table that includes only those columns defined in the subscription-set member. A user-copy table can have the same structure as the source table or it can have a subset of source columns, with or without before images or calculated columns.

  DB2 replication assumes that it is the only application writing to user-copy target tables. Direct changes to user-copy tables by end-users or applications can be overwritten by DB2 replication and can cause the data in the source and target tables to not match. If you need to update both the source and target tables, consider using update-anywhere replication.

Point-in-time
: Read-only target table that includes the columns defined in the subscription-set member and a timestamp column. A point-in-time table can have the same structure as the source table or it can have a subset of source columns, with or without before images or calculated columns.

Base aggregate
: Read-only target table that uses SQL column functions (such as SUM and AVG) to compute summaries of the entire contents of the source table.

  A base-aggregate table summarizes the contents of a source table. A base-aggregate table also includes a timestamp of when the Apply program performed the aggregation. Use a base-aggregate table to track the state of a source table on a regular basis.

Change aggregate
: Read-only target table that uses SQL column functions (such as SUM and

AVG) to compute summaries of the entire contents of recent changes made to the source table, which are stored in the CD table or in an internal CCD table.

A change-aggregate table summarizes the contents of a CD table or in an internal CCD table, rather than the source table. A change-aggregate table also includes two timestamps to mark the time interval for when the changes were captured (written to the CD or CCD table). Use a change-aggregate table to track the changes (UPDATE, INSERT, and DELETE operations) made between replication cycles.

CCD (consistent-change data)
Read-only target table with additional columns for replication control information. These columns include: a log-record number (or journal-record number), an indicator of whether the source table was changed using an SQL INSERT, DELETE, or UPDATE statement, and the log record number and timestamp of the commit statement associated with the insert, delete, or update. You can also optionally include before-image columns and columns from the UOW table.

Replica
Read/write target table for update-anywhere replication. A replica table is the only type of target table that your application programs and users can update directly. Thus, a replica table receives changes from the master table and from local application programs or users. Replica tables can have the same structure as the source table or they can have a subset of source columns, but they do not include any additional replication control columns (such as timestamps). Replica tables are supported only for DB2 databases.

The following sections discuss possible uses for each target type. Each section guides you through the types of target tables that you can use and how you can set the target-table properties to meet your replication needs:

- "Defining read-only target tables"
- "Replicating the net change for a row to the target table" on page 74
- "Defining middle tiers in a multi-tier configuration" on page 75
- "Defining read-write targets (update-anywhere)" on page 77
- "Using an existing table as the target table" on page 79

After you select your target table type, you can use the default settings for the target table, or you can modify the target table properties to meet your replication needs. See "Common properties for all target table types" on page 80 for a complete list of common target table options, their defaults, and an explanation of when you might want to use or change the defaults.

Defining read-only target tables

Target table types: User copy, point-in-time, base aggregate, change aggregate, CCD

Depending on how you want the source data to appear at your target, you can define read-only target tables to contain:

- "A copy of the source table or view" on page 72
- "A history of changes or audit information" on page 73
- "A computed summary of data or changes at the source" on page 72

A copy of the source table or view

Target table types: User copy, point-in-time

Copy of source table: By default, a user copy table will be created as your target type when you define a subscription-set member. Use this default type if you want the target table to match the source table at the time the copy is made. User copy tables do not contain any additional replication-control columns, but they can contain a subset of the rows or columns in the source table or additional columns that are not replicated.

Copy of source table with timestamp: Select point-in-time as your target type if you want to keep track of the time at which changes were applied to the target. A point-in-time target contains the same data as your source table, with an additional timestamp column added to let you know when the Apply program committed each row to the target. The timestamp column is originally null. Point-in-time tables can contain a subset of the rows or columns in the source table or additional columns that are not replicated.

Restriction: DB2 prevents values from being inserted in columns of a DB2 table that are defined AS IDENTITY GENERATED ALWAYS. To avoid this restriction, you can:

- Create the target table without the IDENTITY CLAUSE
- Create the target table with the column AS IDENTITY GENERATED BY DEFAULT

A computed summary of data or changes at the source

Target table types: Base aggregate, change aggregate

Restrictions: Non-DB2 relational targets cannot be aggregate target-table types. Non-DB2 relational sources cannot have aggregate target-table types.

You can create target tables that contain summaries of the entire contents of the source tables or of the most recent changes made to the source table data. For aggregate target-table types, you can define target columns by using aggregate SQL column functions such as COUNT, SUM, MIN, MAX, and AVG. These columns do not contain the original source data; they contain the computed values of the SQL function that you define. The Apply program doesn't create aggregations during full refresh; rows are appended over time as the Apply program processes the set. An advantage of using an aggregate table is that DB2 replication can replicate summary information only rather than each individual row, thus saving both network bandwidth and space in the target table.

Summarizing contents of the source table:

Use a base-aggregate target table to track the state of a source table during each replication cycle. For a base-aggregate target table, the Apply program aggregates (reads and performs calculations) from the source table. A base-aggregate table also includes a timestamp of when the Apply program performed the aggregation.

If a registered source table has only a base-aggregate table as its target, you do not need to capture changes for the source table.

Example: Suppose that you want to know the average number of customers that you have each week. If your source table has a row for each customer, the Apply program would calculate the sum of the number of rows in your source table on a weekly basis and store the results in a base aggregate table. If you perform the

aggregation every week, the target table will have 52 entries that show the number of customers you had for each week for the year.

Summarizing contents of the CD or CCD table:

Use a change-aggregate target table to track the changes (UPDATE, INSERT, and DELETE operations) made between replication cycles at the source table. For a change-aggregate target table, the Apply program aggregates (reads and performs calculations) from the CD or internal CCD table. A change-aggregate table also includes two timestamps to mark the time interval for when the Capture program inserted changes into the CD or CCD table.

Example: Suppose that you want to know how many new customers you gained each week (INSERTs) and how many existing customers you lost (DELETEs). You would count the number of inserted rows and deleted rows in the CD table on a weekly basis and store that number in a change-aggregate table.

Important: If the source table for a subscription-set member is registered for full-refresh only replication, then you cannot have a change aggregate target table, which requires a CD or CCD table at the source.

A history of changes or audit information

Target table type: CCD

You might want to audit the source data or keep a history how the data is used. By using a CCD table as your target type, you can track the history of source changes in various ways, depending on how you define the CCD table. For example, you can track before and after comparisons of the data, when changes occurred, and which user ID made the update to the source table.

To define a read-only target table that keeps a history of your source table, define the target CCD table to include the following attributes:

Noncondensed
: To keep a record of all of the source changes, define the CCD table to be noncondensed, so it stores one row for every change that occurs. Because noncondensed tables contain multiple rows with the same key value, do *not* define a unique index. A noncondensed CCD table holds one row per UPDATE, INSERT, or DELETE operation, thus maintaining a history of the operations performed on the source table. If you capture UPDATE operations as INSERT and DELETE operations (for partitioning key columns), the CCD table will have two rows for each update, a row for the DELETE and a row for the INSERT.

Complete or noncomplete
: You can choose whether you want the CCD table to be complete or noncomplete. Because noncomplete CCD tables do not contain a complete set of source rows initially, create a noncomplete CCD table to keep a history of updates to a source table (the updates since the Apply program began to populate the CCD table).

Include UOW columns
: For improved auditing capability, include the extra columns from the UOW table. If you need more user-oriented identification, columns for the DB2 for z/OS correlation ID and primary authorization ID or the OS/400 job name and user profile are available in the UOW table. For details on which UOW columns you can include in a CCD table, see “Consistent-change data (CCD) table” on page 502.

Replicating the net change for a row to the target table

Target table type: internal CCD

If changes occur frequently at a source table, you can create an internal CCD table to summarize the committed changes that occurred at the source since the last Apply cycle. Because the CD table is constantly in flux when the Capture program appends changes from the log, the local cache of source changes in the CCD acts as a more stable source for your targets.

When the original source table is updated, the Capture program reads the frequent changes in the source's log and adds them to the source's CD table. From that CD table, an Apply program reads the changes in the CD table and populates the internal CCD table. You can define the internal CCD table to contain only the most recent change for each row in the CD table that occurred during the last cycle. Therefore, the CCD table is static between Apply cycles (for the Apply program replicating from the CD table to the CCD table) and thus makes a more stable source for targets. By condensing changes from the source, you can improve overall replication performance by not replicating many updates for the same row to the target table.

Because the Capture program is constantly adding new changes to the CD table, a second Apply program reads changes from the internal CCD table, instead of the CD table, so that it doesn't replicate different changes to different targets and can keep the targets in synch with one another. The second Apply program uses the original source table for full refreshes, and it uses the internal CCD table for change-capture replication.

Recommendations:

- Define a subscription-set member between the source table and the internal CCD table *before* defining other subscription-set members between the source table and other target tables. That way, the Apply program will use the internal CCD table rather than the CD table for replicating changes from the source table. If you define other subscription-set members and begin replication using those members before you define the internal CCD table for the source table, you might have to perform a full refresh for all targets of the source table.
- Combine all internal CCD tables into one subscription set to ensure that all target tables for the source database are in synch with one another.
- Even if you only want a subset of the frequently changing source columns to be applied to other targets, use the default that all registered source columns are replicated to the internal CCD. That way, you can use the internal CCD table as a source for future target tables that might need data from the other registered columns in the original source table. Only columns in the internal CCD table will be available for change-capture replication for any future target.

You use an internal CCD table as an implicit source for replication; you cannot explicitly define it as a replication source. When you add a subscription-set member, you map the original source table (not the internal CCD table) to the target table. An internal CCD table has the following attributes:

Internal
: The CCD table acts as an alternative to the source's CD table. Information about the internal CCD table is stored in the same row as its source table in the register (IBMSNAP_REGISTER) table; an internal CCD table does not have its own row in the register table. The Apply program

automatically replicates changes from an internal CCD table, if one exists, rather than from CD tables. Only one internal CCD table can exist for each replication source.

Restriction: The user table does not include computed columns; therefore, do not include computed columns in the CCD subscriptions.

Local The CCD table is in the same database as the source table.

Noncomplete
Because the Apply program uses the original source table for full refreshes and not the internal CCD, the CCD is noncomplete because the subsequent target will already have an initial copy of all the source rows.

Condensed
The internal CCD is condensed, meaning that table contains one row for every key value, so that the Apply program applies the most recent change for each row in the CCD table, instead of applying a row for every change.

No UOW columns
Internal CCD tables do not support additional UOW table columns. You cannot use an internal CCD table if you already defined a target CCD table that includes UOW columns.

Important for update-anywhere: If you define an internal CCD table, the Apply program ignores it when processing a subscription set with a replica as a target, and it applies changes to the replica from the master source's CD table.

Defining middle tiers in a multi-tier configuration

Target table type: CCD

The basic replication model is a two-tier model, with a single source and one or more targets; but you can set up configurations with three (or more) tiers. A multi-tier configuration has a source table and a target table, and then that target table acts as a source to other target tables.

One reason to set up a multi-tier replication environment is to provide stable sources for the third-tier targets. Because you can collect changes from tier 1 in CCD tables at tier 2, you can control how often you replicate changes to each tier and reduce the number of changes replicated to the target (tier 3). You can also avoid many of the database connections to your source system, thus moving the connection cost to the second tier.

For example, in a three-tier model, the first tier (tier 1) is the source database, the second tier (tier 2) is the target for tier 1. Tier 2 is also a source for a third tier of targets (tier 3), and can distribute changes to one or many tier-3 databases. When you have more than two tiers in your replication configuration, the middle tiers, which act as both sources and targets, are CCD tables.

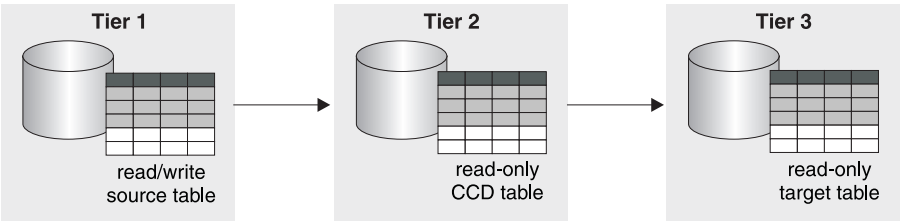


Figure 6. Three-tier replication model. You can replicate data from a source table to a target table, and then from that table to another target table.

Restriction:

You cannot use a non-DB2 relational table or a CCD table in a non-DB2 relational database as a middle tier in multi-tier configurations.

Procedure:

This procedure also applies for replica tables. CCD tables are usually used for read-only replication, but replica tables are used for update-anywhere replication.

To set up multi-tier replication, so that your target table acts as a source to subsequent targets:

1. Register the source table (tier 1) for replication. For information on how to register a table for replication, see "Registering DB2 tables as sources" on page 35.

   The Capture program for this source captures changes that occur at tier 1 and stores them in tier 1's CD table.
2. Create a subscription set between the source server and the target server (for tier 2). For information on how to create a subscription set, see "Creating subscription sets" on page 59.

   The Apply program for this subscription set applies changes from tier 1 to the CCD table at tier 2.
3. Define a subscription-set member mapping the source table (tier 1) and a CCD target table (tier 2). For information on how to define subscription-set members, see "Mapping source tables and views to target tables and views within a subscription set" on page 67.

   When defining the target table for this member, select for the target table to be a CCD table with the following attributes:

   External registered source
   : You must define it as an external target table and register the table so it can act as a source for the subsequent tier. Like other registered sources, an external CCD table has its own row in the register (IBMSNAP_REGISTER) table. External CCD tables that also act as sources can be populated only by a single source table.

     You must register all external CCD tables in a subscription set using the same Capture schema.

   Complete
   : You must use a complete CCD table because the Apply program will use this table to perform both full refresh and change-capture replication for the subsequent tier.

   Condensed
   : Use a condensed CCD, meaning that table contains one row for every

key value, to ensure that only the most recent changes are replicated to the subsequent tier. The Apply program applies the most recent change for each row in the CCD table, instead of applying a row for every change. Because condensed tables require unique key values for each row, you *must* define a unique index.

4. Because the CCD table is registered, create the Capture control tables in the middle-tier database, if they do not already exist.
5. Create a subscription set between the tier 2 server that contains registered CCD table and the subsequent target server (for tier 3). For information on how to create a subscription set, see "Creating subscription sets" on page 59.

   The Apply program for this set applies changes from the CCD table to the target tables in the subsequent tier. The Apply program uses the CCD table for both full refresh and change-capture replication. Usually, you use a different Apply qualifier than the one used to populate the CCD, but you can use the same one.
6. Define a subscription-set member mapping the CCD source table (tier 2) and the subsequent target table (tier 3). For information on how to define subscription-set members, see "Mapping source tables and views to target tables and views within a subscription set" on page 67.

   You can set up multiple members with target tables that subscribe to this CCD source table. If this is the final tier in your multi-tier configuration, then the target table can be any type. However, if you plan to have more than three tiers, define the tier-3 target table as specified in step 3, and repeat steps 4 through 5 to add subsequent tiers.

Important: If a full refresh occurs on the external CCD (the middle tier), then the Apply programs for all subsequent tiers that use that external CCD as a source will perform full refreshes. This is called a *cascade full refresh*.

Defining read-write targets (update-anywhere)

Target table type: Replica

In update-anywhere replication, changes at the master source table are replicated to dependent target tables that are of type replica, and changes at the replica tables can be replicated back to the master source table. In update-anywhere replication, the master table and its replicas are read-write tables that all act as both sources and targets.

Prerequisites:

The following conditions must exist for update-anywhere replication:

- You must use declarative referential-integrity constraints because no single application program updates both master and replica tables. Referential-integrity violations cannot be detected in application logic.
- You must include all referential constraints that exist among the master tables in the replica tables to prevent referential-integrity violations. If you omit some referential constraints, an update made to a replica table could cause an referential-integrity violation when it is replicated to the master table. The administration tools do not copy referential-constraint definitions from a source table to target tables, nor can they generate new constraints.
- To bypass referential-integrity checking during full refresh, you must use the ASNLOAD exit routine.

Restrictions:

- Replica target table types are not supported in a remote journal configuration.
- You cannot use CCD tables as sources or targets in update-anywhere replication.
- To allow columns of LOB data type to participate in update-anywhere replication, the CONFLICT_LEVEL in the register table must be set to 0.
- Columns of DATALINK data type cannot participate in update-anywhere replication, except when you register the source table with no conflict detection.
- Non-DB2 databases cannot have replica target-table types and, therefore, cannot participate in update-anywhere replication.

Procedure:

To set up an update-anywhere configuration between a master table and one or more replica tables (where each replica table is in a separate database):

1. Because the Capture program will capture changes for each replica table, create the Capture control tables in each database that will contain a replica table, if they do not already exist.
2. Register the source table (the master table) for replication. For information on how to register a table for replication, see "Registering DB2 tables as sources" on page 35.

   The Capture program for this source captures changes at the master table and stores them in the master's CD table.
3. Create a subscription set between the master database and the target database that will contain the one or more replicas. For information on how to create a subscription set, see "Creating subscription sets" on page 59.

   If all replica tables are in the same database and all master tables are in another database, you need only one subscription set. If the replica tables are in multiple databases, you need as many subscription sets as you have replica databases.
4. Define a subscription-set member for each mapping between each master table and its associated replica table. For information on how to define subscription-set members, see "Mapping source tables and views to target tables and views within a subscription set" on page 67.

   In this configuration, there is only one Apply program, which typically runs at the server that contains the replica tables. The Apply program for this set pulls the changes from the master's CD table and applies them to the replica tables. The Apply program also pushes changes from the replica table's CD table and applies them to the master table.

   Important: Because the master table and replica tables in update-anywhere configurations replicate data back and forth to one another, replica target tables should contain the same columns as the source table. You can create a replica target that contains a subset of the columns in the master table only if the missing columns are defined as nullable or NOT NULL WITH DEFAULT at the master site, but you should not add new columns or rename columns at the replica.
5. Define source properties for the replica table.

   When you create a subscription-set member with a replica table, DB2 replication automatically registers the replica table as a replication source. Because replica target tables act as sources, they have properties that you can set in addition to the common target table properties, which determine how the Capture program handles changes to the replica. There are two properties, however, that are inherited from the master table and cannot be changed for

the replica table: the conflict-detection level and whether full refreshes are disabled. The Capture program for this source captures changes at the replica table and stores them in the replica's CD table. See "Registration options for source tables" on page 38 for a complete list of registration options, their defaults, and an explanation of when you might want to use or change the defaults.

Important: Even though the master and replica act as both sources and targets, full-refresh copying occurs only from the master to the replica, not from replica to master.

To prevent conflicts, you must make the target key for the replica tables the same as the master source table's primary key or unique index. Because the master table can update the replicas and the replicas can update the master, there is a potential for conflicts to occur if an update is made to a row in the master table and a different update is made to the same row in one or more replica tables between Apply cycles (so that the changes are in the master CD table and the replica CD table). A replica table inherits the level of conflict detection from the master source table or view. It is best to design your application so that a conflict can never occur when data is replicated from the master to all of the replica tables. When you registered the master source, you had three levels of conflict detection to choose from. For more information on selecting a level of conflict detection and on how to deal with conflicts that occur (if you selected either standard or enhanced conflict detection), see "Setting conflict detection (update-anywhere replication)" on page 49.

If you defined referential integrity constraints for the source table, you must define the same referential integrity constraints for the replica table to prevent integrity violations. If a referential-integrity violation occurs, the subscription cycle is automatically retried.

Using an existing table as the target table

You can use a previously-defined DB2 table as the target table in a subscription set. That is, you can define a subscription-set member to include a target table that you defined outside of DB2 replication. Such a user-defined target table can be any of the valid target-table types for replication (user copy, point in time, base or change aggregate, CCD, or replica) as long as the structure of the table is valid. For example, a user-defined point-in-time table must include a column of type TIMESTAMP called IBMSNAP_LOGMARKER.

Requirements:

- If the subscription-set member definition contains fewer columns than are in the existing target table, the target-table columns that are not involved in replication must allow nulls or be defined as NOT NULL WITH DEFAULT.
- There must be a unique index for point-in-time, user copy, replica, and condensed CCD tables. When you define the subscription-set member using the existing target table, you can use the existing unique index or specify a new one.

Restrictions:

- A subscription-set member definition cannot contain more columns than are in the existing target table.
- If you are using the Replication Center, you cannot add a column to a subscription-set member if that column does not already exist in the target table.

DB2 replication checks for inconsistencies between your existing target table and the subscription-set member definition.

Important for multi-tier: If you want to set up a multi-tier configuration with a source table as tier 1, a CCD table as tier 2, and an existing table as tier 3, define the CCD table to match the attributes specified for the existing target table when defining the subscription-set member between tier 1 and tier 2. Then define a subscription-set member for the existing target table in which the CCD table is the source table.

Common properties for all target table types

This section discusses the common properties that you can set when creating a target table, regardless of type. You can modify properties for your target table or view based on the type of replication that you want. The following sections explain the common characteristics that you can define for how the source data maps to the target tables:

- "Source columns that you want applied to the target"
- "Source rows that you want applied to the target"
- "How source columns map to target columns" on page 81
- "Target key" on page 82
- "How the Apply program updates the target key columns with the target-key change option" on page 83

Source columns that you want applied to the target

Default: all registered source columns are replicated to the target

In some replication scenarios, you might not want to replicate all columns to the target table, or the target table might not support all data types defined for the source table. You can define a column (vertical) subset that has fewer columns than your source table.

By default, your target table contains all the registered columns from the source table, except LOB and DATALINK columns. If you don't want the target table to contain all of the columns that exist in the source table, select *only* those source columns that you want to replicate to the target table. The registered columns in the source table that you do not select are still available for other subscription-set members, but are not included for the current source-to-target mapping.

You can also add calculated columns to a target table. These columns can be defined by SQL scalar functions, such as SUBSTR, or they can be derived columns, such as the division of the value of column A by the value of column B (colA/colB). These calculated columns can refer to any columns from the source table.

Source rows that you want applied to the target

Default: all source rows are replicated to the target

By default, your target table contains all the rows in the source table. For some replication scenarios, you might not want to replicate all rows from the source table to the target table, or you might want to replicate source rows containing different sorts of data to different target tables. You can define a row (horizontal) subset that contains rows matching a certain condition (an SQL WHERE clause). The SQL predicate can contain ordinary or delimited identifiers. See the *DB2 SQL Reference* for more information about WHERE clauses.

Examples:

- Assume that your target table is an operational data store for one of your company's operational divisions. You can define a WHERE clause in the subscription-set member to replicate all rows for the division (or all departments in the division) from the source table to the target table.
- Assume that you have several target tables in the same database. You can define a WHERE clause in one subscription-set member to replicate all LOB columns (plus the primary-key column) to one target table, and you can define a WHERE clause in another subscription-set member to replicate all other columns to a separate target table. Thus, your target database can have all of the data from the source table, but denormalize the source table in the target database to adjust query performance for a data warehouse.

Row predicate restrictions:

- Do not type `WHERE` in the clause; it is implied. Type `WHERE` in the clause only for subselect statements.
- Do not end the clause with a semicolon (;).
- If you want to use before-image columns, computed columns, or IBMSNAP columns to subset or filter your data, see "Subsetting data during subscription" on page 95.
- If your WHERE clause contains the Boolean expression OR, enclose the predicate in parentheses; for example, `(COL1=X OR COL2=Y)`.
- If the target table is a change aggregate table and contains before-image columns, you must include the before-image columns in a GROUP BY clause.

The following examples show WHERE clauses that you can use to filter rows of the target table. These examples are very general and are designed for you to use as a model.

- WHERE clause specifying rows with specific values

  To copy only the rows that contain a specific value, such as MGR for employees that are managers, use a WHERE clause like:

  ```
  EMPLOYEE = 'MGR'
  ```

- WHERE clause specifying rows with a range of values

  To copy only the rows within a range, such as employee numbers between 5000 and 7000 to the target table, use a WHERE clause like:

  ```
  EMPID BETWEEN 5000 AND 7000
  ```

How source columns map to target columns

Default: source column name maps to same target column name (if target table does not yet exist)

By default, the column names in the target table (if it does not yet exist) will match the column names in the source table, and the data value in a source column will be replicated to the target column with the same name. You can change the names of all columns in your target tables except the replication control columns (which begin with IBMSNAP or IBMQSQ). If the target table exists, the Replication Center will map the columns by name.

Target table columns can have different lengths than source columns. If the target column is shorter than the source column, you can use an expression in the subscription-set member to map the characters from the longer column to the shorter column, or register a view that includes the expression. For example, if the source column is char(12) and the target column is char(4), you can use the following expression to truncate the values from COL1 during replication:

```
substr(col1, 1,4)
```

If the target column name is longer, pad the target column name with blanks.

If you are mapping a DB2 table to a non-DB2 relational table with an existing nickname for the non-DB2 relational table, the data types of some columns might not be compatible. If the data types of the source columns are not compatible with the data types in the target columns, you can modify the data type at the target to make it compatible with the source:

- You can add calculated columns to adjust the data types from the source to match the required data type for the target.
- You can alter the nickname for a non-DB2 relational target table to change the data-type conversions.

Some restrictions exist for mapping long variable characters (LONG VARCHAR) in DB2 Universal Database to both z/OS and OS/400. See "General data restrictions for replication" on page 85 for details on data type restrictions.

Example: You want to replicate data from a DB2 source table with a DB2 column of data type DATE to an Oracle target table with an Oracle column of data type DATE.

Table 5. Mapping a DB2 DATE column to an Oracle DATE column

DB2 Column	Nickname Data Mapping	Oracle Column
A_DATE DATE	A_DATE TIMESTAMP A_DATE DATE	A_DATE DATE

The Oracle target table is created with an Oracle data type of DATE (which can contain both date and timestamp data). The initial nickname for an Oracle DATE data type in a federated database maps the DB2 data type as a TIMESTAMP. The DB2 Replication Center and the OS/400 system commands for replication alter the nickname data type to DATE, so that a DATE is replicated to Oracle and not a TIMESTAMP.

When you are creating a target table using the Replication Center, you can rename columns at the target regardless of the target-table type. Also, you can change column attributes (data type, length, scale, precision, and whether it is nullable) where the attributes are compatible. You cannot use the Replication Center to rename columns of existing target tables. If the source and target columns do not match, you can either use the Replication Center to map the columns from the source to the target, or you can create a view of the target table that contains a match to the source column names.

Target key

Default index name: The default name comes from the target object profile for the target server, if there is one. If you have not set this profile, the default is IX plus the name of target table. For example, if the name of your target table is TGEMPLOYEE, the name of your target table index defaults to IXTGEMPLOYEE.

When a condensed target table is involved in change-capture replication, the Apply program requires it to have a primary key or unique index, which is called the *target key*. You can choose which columns you want to use as the unique index for your target table. The following types of target tables are condensed and require a target key:

- User copy

- Point-in-time
- Replica
- Condensed CCD

If you are creating a new target table, you can use the default index name and schema or change the defaults to match your naming conventions.

To create a unique index for a new target table, you have two options:

- Specify the columns that you want as the unique index for the target table.
- Have DB2 replication select a unique index for you.

  If you do not select columns for the unique index, DB2 replication checks the source table for one of the following definitions, in the following order:

  1. A primary key
  2. A unique constraint
  3. A unique index

  If DB2 replication finds one of these definitions for the source table, and the associated columns are registered and part of the target table, DB2 replication uses the source table's primary key (or unique index or RRN) as the target key. In the case of a unique constraint, DB2 replication creates a unique index for the target table using the constraint columns.

  For an OS/400 source table that does not have a primary key or unique index, modify the registration for that table to use the relative record number (RRN) as a uniqueness factor. When you define the subscription-set member, specify the RRN column as the unique index for the target table. See "Using relative record numbers (RRN) instead of primary keys (OS/400)" on page 51 for details on defining an RRN for an OS/400 source table.

  For target tables on OS/400 systems that use the RRN as the target key, you should run the Apply program on an OS/400 system to replicate to these target tables.

For existing target tables, you must select the unique index. You can select one of the following options:

- Use an index that already exists for the target table.

  To use an existing index, select the columns that represent the index in the Replication Center. If the Replication Center finds an exact match then it only sets a target key for the Apply program to use, otherwise it creates the unique index and sets a target key for the Apply program to use.
- Create another index for the target table.

  The unique index will be created if it does not already exist, and the target key will be set for the Apply program to use.

Important: If you select a key for the target table that includes columns that can be updated at the source table, you must instruct the Apply program to make special updates to the target key columns. See "How the Apply program updates the target key columns with the target-key change option" for more information.

How the Apply program updates the target key columns with the target-key change option

Restrictions:

- You cannot use the target-key-change option for source tables that are registered to capture updates as delete/insert pairs.

- You cannot map an expression in a source table to a key column in a target table if the Apply program updates the target table based on the before images of the target key column (that is, if the TARGET_KEY_CHG column of the IBMSNAP_SUBS_MEMBR table has a value of `Y` for that target table).

If you choose the target-key change option when you define the subscription-set member, then the Apply program makes special updates to the target key columns when the target key changes. In order for the Apply program to make these special updates, the columns that are in the source table that are part of the target-key columns for the target table must be registered with the before-image columns in the CD (or CCD) table. If you did not define the source registration to capture the before-image values of the columns that make up the target key, then you must alter your registration to include them before subscribing to a target table with a different key.

After you ensure that the before-image values of the target key columns are in the CD (or CCD) table, select the subscription-set member option for the Apply program to use the before-image values when updating target key columns.

If you do not specify for the Apply program to use the before-image values when updating target key columns, DB2 replication will not replicate data correctly when you update the columns in the source table that are part of the target key. The Apply program attempts to update the row in the target table with the new value, but it does not find the new key value in the target table to update it. The Apply program then converts the update to an INSERT and inserts the new key value in the target table. In this case, the old row with the old key value remains in the target table (and is unnecessary). When you specify that you want changes to target key columns to be processed using before-image values, the Apply program is able to find the row with the old key value, and update the row using the new values. For example, if the *target_key_chg* variable is set to **N**, the SQL statement for the update operation is:

```
UPDATE targettable SET <non-key columns>= after-image values
WHERE <key columns> = after-image values
```

If the *target_key_chg* variable is set to **Y**, the SQL statement for the update operation is:

```
UPDATE targettable SET <all columns> = after-image values
WHERE <key columns> = before-image values
```

Related concepts:

- Chapter 15, “Using the Replication Center for SQL replication,” on page 219

Related tasks:

- Chapter 3, “Registering tables and views as SQL replication sources,” on page 35
- Chapter 6, “Subsetting data in an SQL replication environment,” on page 93
- Appendix A, “UNICODE and ASCII encoding schemes for SQL replication (z/OS),” on page 507

Related reference:

- “ADDDPRSUBM: Adding a DPR subscription-set member (OS/400)” on page 343
- “ADDDPRSUB: Adding a DPR subscription set (OS/400)” on page 327
- “Consistent-change data (CCD) table” on page 502

Chapter 5. Replicating special data types in SQL replication

When you replicate special data types, such as LOB, DATALINK, ROWID, or non-DB2 data types, you should be aware of certain conditions and restrictions. In some cases, you might have to perform additional setup steps to get DB2 replication to work with these data types. This chapter discusses these conditions and restrictions and includes the following sections:

- "General data restrictions for replication"
- "Replicating large objects"
- "Replicating DATALINK values" on page 86

General data restrictions for replication

Currently, DB2 replication has specific restrictions for certain data types.

DB2 replication *cannot* replicate the following data types under any circumstances:

- LOB columns from non-DB2 relational sources
- Any column on which any of the following procedures is defined:
  - EDITPROC
  - FIELDPROC
  - VALIDPROC

DB2 replication *can* replicate the following data types under certain circumstances:

- Long variable graphic (LONG VARGRAPHIC) data if the source and target tables reside in DB2 for z/OS.
- Long variable character (LONG VARCHAR) data requires either that the source database tables be in DB2 for z/OS or both the source and target tables be in DB2 Universal Database (for Windows, Linux, and UNIX). See the Alter Table section of the *DB2 Universal Database SQL Reference* to learn how to enable LONG VARCHAR data.

DB2 replication *cannot* replicate a table that contains abstract data types.

DB2 replication *can* replicate tables with spatial data type columns but *cannot* replicate the actual spatial data type columns.

User-defined data types (distinct data types in DB2 Universal Database) are converted to the base data type in the change-data (CD) table before replication. In addition, if DB2 replication creates the target table as part of the subscription-set member definition, user-defined types are converted to the base data type in the target table as well as in the CD table.

Replicating large objects

DB2 Universal Database supports large object (LOB) data types, which include: binary LOB (BLOB), character LOB (CLOB), and double-byte character LOB (DBCLOB). This section refers to all of these types as LOB data.

The Capture program reads the LOB descriptor in the log records to determine if any data in the LOB column has changed and thus should be replicated, but does

not copy the LOB data to the change-data (CD) tables. When a LOB column changes, the Capture program sets an indicator in the CD table. When the Apply program reads this indicator, the Apply program then copies the entire LOB column (not just the changed portions of LOB columns) directly from the source table to the target table.

Because a LOB column can contain up to two gigabytes of data, you must ensure that you have sufficient network bandwidth for the Apply program. Likewise, your target tables must have sufficient disk space to accommodate LOB data.

Restrictions:

- The Apply program always copies the most current version of a LOB column directly from the source table (not the CD table), even if that column is more current than other columns in the CD table. Therefore, if the LOB column in the target row changes, it is possible that this LOB column could be inconsistent with the rest of the data in that target row. To reduce this possibility of inconsistent data in the target row, ensure that the interval between the Apply cycles is as short as practical for your application.
- You can replicate 10 LOB columns or fewer per table. If you register a table with more than 10 LOB columns, the Apply program returns an error message. The Replication Center returns an error message if you attempt to register more than 10 LOB columns per table.
- You can copy LOB data to replica tables provided that conflict detection is disabled.
- To copy LOB data between DB2 for OS/390 Version 6 (or later) and DB2 Universal Database (for any other operating system), you need DB2 Connect 7 or later.
- You cannot refer to LOB data using nicknames.
- Before-image values for LOB, DATALINK, or ROWID columns are not supported.
- Replication is not supported for DB2 Extenders™ for Text, Audio, Video, Image, or other extenders where additional control files associated with the extender's LOB column data are maintained outside of the database.
- DB2 replication can replicate a full LOB only; it cannot replicate parts of a LOB.
- You cannot replicate LOB columns if you use a remote journal setup in your replication environment on OS/400.

Replicating DATALINK values

Accessing large files (such as multimedia data) over a remote network can be inefficient and costly. You can access and replicate unstructured files more quickly by using the DATALINK data type to represent data that is stored in external file systems.

DB2 Universal Database supports the DATALINK data type that allows the database to manage the access control, referential integrity, and recovery of these large and unstructured files. DB2 Universal Database supports DATALINK values on the following operating systems:

- AIX
- Solaris™ Operating Environments
- Windows
- OS/400

A DATALINK column value contains a Uniform Resource Locator (URL) that points to the location of an external file. DB2 replication uses the following components when replicating DATALINK column values and the files that they reference:

ASNDLCOPY exit routine
Maps the URL on the source file system to the URL on the target file system and then connects to the appropriate file-copy daemon to replicate the external file to which the URL points.

Data Links Manager replication daemon (DLFM_ASNCOPYD)
Works with the ASNDLCOPY exit routine to copy the files that are referenced by DATALINK column values. The DLFM_ASNCOPYD daemon is part of DB2 Data Links Manager Version 8. You can use this daemon on AIX, Solaris™ Operating Environment, and Windows operating systems.

ASNDLCOPYD daemon
Works with the ASNDLCOPY exit routine to copy the files that are referenced by DATALINK column values and is shipped with DB2 for iSeries. Use the ASNDLCOPYD daemon on OS/400 and optionally on other operating systems.

When the Apply program reads data with a data type of DATALINK, the Apply program places reference data in the spill file and also places the URL of the updated file into an input file.

The Apply program then invokes the ASNDLCOPY exit routine. This ASNDLCOPY exit routine ensures that the physical file exists on the source file system, maps the URL to its corresponding file on the target file system, stores this target file location in a result file, and then connects to the appropriate file-copy daemon (DLFM_ASNCOPYD, ASNDLCOPYD, or FTP) to copy the external file from the source file system to the target file system.

Recommendation: Use a separate subscription set for DATALINK columns because the Apply program waits for the ASNDLCOPY routine to complete its processing before the Apply program completes replication of the subscription set. Any failures in copying the external files will cause replication of the entire subscription set to fail. If the subscription set fails, the Apply program will not deactivate the subscription set but will process the subscription set again during the next Apply cycle.

On Linux, UNIX, and Windows: Start the Apply program with the **loadxit** parameter set to `y` to invoke the ASNLOAD exit routine. The ASNLOAD exit routine copies external files (to which the DATALINK values point) during a full refresh. See “Refreshing target tables using the ASNLOAD exit routine” on page 135 for more information.

On OS/400: Modify the ASNLOAD exit routine to call the ASNDLCP exit routine to enable the Apply program to copy external files during a full refresh. See “Refreshing target tables using the ASNLOAD exit routine” on page 135 for more information.

Important: Because external files can be very large, you must ensure that you have sufficient network bandwidth for both the Apply program and the file-transfer mechanism that you use to copy these files. Likewise, your target system must have sufficient disk space to accommodate these files.

Restrictions:

- You cannot replicate DATALINK columns between DB2 databases on OS/400 and DB2 databases on other operating systems.
- On the OS/400 operating system, there is no support for the replication of the "comment" attribute of DATALINK values.
- If you use update-anywhere replication with DATALINK columns, you must specify **None** for the conflict-detection level to turn off the conflict detection for both the DATALINK columns and the other columns in the same subscription set. DB2 replication does not check update conflicts of external files referenced by DATALINK columns.
- Before-image values for DATALINK columns are not supported.
- Target tables that are base-aggregate or change-aggregate tables cannot support DATALINK columns.
- When replicating data in consistent-change data (CCD) tables, the following restrictions apply:
  - Internal CCD tables can contain DATALINK indicators (VARCHAR type character strings that contain information about the associated URLs) but not DATALINK values. The Apply program does not invoke the ASNDLCOPY exit routine when replicating data in these table types.
  - Condensed external CCD tables can contain DATALINK columns.
  - Noncondensed CCD target tables cannot contain any DATALINK columns.

The following sections discuss the user exit routine and the file-copy daemons that the Apply programs use (depending on the operating system) to replicate both the DATALINK values and the external file to which the URL points to the target system:

- "Setting up and using the ASNDLCOPY exit routine"
- "Setting up and using DLFM_ASNCOPYD (Linux, UNIX, Windows)" on page 90
- "Setting up and using ASNDLCOPYD (OS/400)" on page 91

Setting up and using the ASNDLCOPY exit routine

When a subscription set is ready to be replicated, the Apply program identifies the applicable rows in the change-data (CD) table. If any DATALINK column values are found, the Apply program places the URLs of the updated files into the input file. The Apply program then invokes the ASNDLCOPY exit routine, which reads this input file and maps each DATALINK source file location to its corresponding target file location. Then, the ASNDLCOPY exit routine connects to the file-copy daemon and replicates the external file to which the URL points from the source file system to the newly mapped target file system location.

When the ASNDLCOPY routine completes, it passes a return code to the Apply program. A nonzero return code tells the Apply program that replication failed for one or more of the files; in this case, the Apply program issues a message, skips the current subscription set, and processes the next subscription set. A zero return code tells the Apply program that replication was successful.

You can use the source code for the ASNDLCOPY exit routine, and modify the sample program (which is called ASNDLCOPY.smp and is located in the \sqllib\samples\repl directory) to meet the requirements of your system. The sample program contains the following configuration files:

ASNDLSRVMAP
Maps the source URL to the target URL.

Example: http://source.com/file to http://target.com/file

ASNDLUSER
Contains the logon and address location information used when connecting to source and target file systems.

ASNDLPARM

Contains operational parameters used to control the function of the ASNDLCOPY exit routine. These parameters include the **REPLACE_FILE** parameter, which is used to replicate a source file to a different target file location, and the **PRESERVE_MODTIME** parameter, which is used to preserve the last modification time of the files you are replicating. ASNDLPARM is an optional configuration file that is used only on Linux, UNIX, and Windows operating systems.

You can configure your own exit routine to replicate the external files, but you must name the program ASNDLCOPY. Place the configuration files in the current execution path of the Apply program.

See the PROLOG section of the sample program in the \sqllib\samples\repl directory for information on how to set up the configuration files and to modify this exit routine.

Procedure:

To use the ASNDLCOPY exit routine:

1. Modify the ASNDLCOPY routine, if necessary, to meet the requirements of your site.

   If you turn on the trace option in the Apply program, the ASNDLCOPY routine creates two files: a log file and a trace file. The log file has the following name:

   ASNDL*ApplyQualSetNameSrcSrvrTgtSrvr*.LOG

   where *ApplyQual* is the Apply qualifier, *SetName* is the subscription-set name, *SrcSrvr* is the source-server name, and *TgtSrvr* is the target-server name. The log file contains all messages generated by the ASNDLCOPY routine. The trace file has the following name:

   ASNDL*ApplyQualSetNameSrcSrvrTgtSrvr*.TRC

   The trace file contains any trace information generated by the ASNDLCOPY routine.
2. Configure the ASNDLUSER, ASNDLSRVMAP, and ASNDLPARM configuration files as necessary.

   On Linux, UNIX, and Windows: If the **REPLACE_FILE** parameter is set to YES (the default) in the ASNDLPARM file and the target file already exists in the target directory, the ASNDLCOPY exit routine replicates the source file contents to a different target system file. The ASNDLCOPY exit routine copies the content of the source file directly into a temporary file, which is given a name that is equal to the source file name with an added suffix of *new*. (You can change this suffix in the ASNDLPARM file.) The Apply program then receives the URL of the original target file and the URL of the temporary file from the result file. When the Apply program propagates changes to the target table, DB2 renames the temporary file to the file name in the original target URL as the replication transaction is committed.

3. If you modified the ASNDLCOPY exit routine, compile the program and place the executable in the appropriate directory.

Because the Apply program calls the ASNDONE exit routine after subscription processing completes regardless of success or failure, you can use the routine to perform any necessary clean up if the ASNDLCOPY routine fails to replicate any external files.

Setting up and using DLFM_ASNCOPYD (Linux, UNIX, Windows)

If you installed DB2 Data Links Manager Version 8, you can use the Data Links Manager replication daemon (DLFM_ASNCOPYD) to copy the files that are referenced by the DATALINK data type.

After the ASNDLCOPY exit routine maps the source and target URLs, this exit routine connects to a daemon to copy the files. You can configure the ASNDLUSER configuration file, specifying the address and port number needed to connect to the file-copy daemon that you want to use. You can use any FTP daemon or the DLFM_ASNCOPYD file-copy daemon.

Both the FTP and the DLFM_ASNCOPYD daemons copy external files from the source file system to the target file system. However, the DLFM_ASNCOPYD file-copy daemon provides additional functionality:

- Allows retrieval of a particular version of a file that is referenced by a DATALINK column defined as RECOVERY YES.
- Allows retrieval of files referenced by DATALINK columns defined as READ PERMISSION DB depending on the access privilege of the user.
- Provides the ability to preserve the last modification time of replicated files.

Restrictions for DLFM_ASNCOPYD:

To copy your replicated files with DLFM_ASNCOPYD, you must use DB2 Data Links Manager Version 8 with DB2 Universal Database Version 8.

You can use the DLFM_ASNCOPYD file-copy daemon with only the following operating systems: AIX, Solaris™ Operating Environments, and Windows.

Restriction for Solaris™ Operating Environments for FTP:

If you are replicating DATALINK column values on Solaris™ Operating Environments and are using the FTP daemon to copy the files, you must use FTP daemons that support the MDTM (modtime) command. The FTP daemons that run in the source and the target file systems must support MDTM, which displays the last modification time of a given file. If you are using Version 2.6 of the Solaris™ Operating Environment, or any other version that does not include FTP support for MDTM, you need additional software such as WU-FTPD.

Procedure:

To set up the DLFM_ASNCOPYD file-copy daemon:

1. Identify the users who require a connection to this file-copy daemon.
2. Grant authority to the users to access files based on the directory where these files are located.

3. Verify that the DLFM_ASNCOPYD daemon is enabled and that the correct port number is specified.

   This port number must match the port number that is specified in the ASNDLUSER configuration file.

For more information, see *DB2 Data Links Manager Quick Beginnings* and *DB2 Data Links Manager Administration Guide and Reference*.

The Data Links File Manager archives a new version of a source file with a DATALINK column defined as RECOVERY YES each time an application links to the file through a standard SQL operation. When the Capture program captures changes to a row with a DATALINK column defined as RECOVERY YES, the Capture program records the version number of the file and places that version number in the CD table. The Apply program reads the data changes from the CD table along with the version number and passes the URLs of the new DATALINK column values and the version number to the ASNDLCOPY exit routine. When the ASNDLCOPY exit routine connects to the DLFM_ASNCOPYD daemon, this file-copy daemon retrieves a consistent version of the external file.

Even if a more recent version of the file exists on the source system, the Data Links File Manager provides the version of the file that is consistent with the version captured in the CD table. Therefore, the target server cannot have a version that the Capture program has not yet captured in the log.

Setting up and using ASNDLCOPYD (OS/400)

ASNDLCOPYD is the daemon that enables authorized users to retrieve files from an OS/400 source server to an OS/400 target server after the ASNDLCOPY exit routine maps the source and target URLs. After the ASNDLCOPY exit routine maps the source and target files, it connects to the ASNDLCOPYD daemon to retrieve the files. The ASNDLCOPYD file-copy daemon is similar to a FTP daemon but provides the following functions when replicating DATALINK values:

- A command for extracting file information (such as file size and last modification time)
- A command for retrieving the content of a particular file

You can configure the ASNDLCOPY exit routine to connect to the ASNDLCOPYD file-copy daemon to replicate a DATALINK column that is defined as READ PERMISSION DB.

You can find the ASNDLCOPYD sample file in library QDP4, source file QCSRC, member ASNDLCPD. The sample file builds three programs:

ASNDLCOPYD
: The main parent program and file-copy daemon.

ASNCHILD
: The program that coordinates connections from the client to the ASNDLCOPYD daemon. ASNCHILD is part of the ASNDLCOPYD daemon, which spawns a new ASNCHILD process for each request from the client.

ASNDLCFG
: A configuration program for adding and removing user IDs and for changing user ID passwords.

Note: If you are currently using an ASNDLCOPYD file-copy daemon under DB2 Version 7 on OS/400 or other operating system, you can continue to use this daemon with DB2 Version 8.

Prerequisite:

You must have root (administrator) authority to run the ASNDLCOPYD daemon.

Procedure:

To use the ASNDLCOPYD file-copy daemon:

1. Access the ASNDLCOPYD sample program in library QDP4, source file QCSRC, member ASNDLCPD.
2. Modify the sample program to meet the requirements of your site.
3. Build the program daemon.
   a. Build the base module:
      ```
      CRTCMOD MODULE(libraryname/ASNDLCPD) SRCFILE(QDP4/QCSRC)
                    DBGVIEW(*SOURCE) SYSIFCOPT(*ALL)
      ```
   b. Build the child program (ASNCHILD):
      ```
      CRTPGM PGM(libraryname/ASNCHILD) MODULE(libraryname/ASNDLCPD)
      ```
   c. Build the parent program (ASNDLCOPYD):
      ```
      CRTPGM PGM(libraryname/ASNDLCOPYD) MODULE(libraryname/ASNDLCPD)
      ```
   d. Build the configuration program (ASNDLCFG):
      ```
      CRTPGM PGM(libraryname/ASNDLCFG) MODULE(libraryname/ASNDLCPD)
      ```
   where *libraryname* is any existing library name. See the PROLOG section of the sample program for more information.
4. Place the executables into the QDP4 library.
5. Modify the configuration files to meet the requirements of your site.
6. Start the ASNDLCOPYD daemon with administrator authority and superuser access. Specify both the port number and the directory that contains the configuration files.

The ASNDLCOPYD file-copy daemon creates a log file for all the messages generated by the ASNDLCOPYD program. This log file has the following name: ASNDLCOPYD*YYYYMMDDHHMMSS*.LOG, where *YYYYMMDDHHMMSS* is the time that the daemon started running.

On OS/400, DB2 replication always replicates the most recent version of an external file that is referenced by a DATALINK column value.

Related tasks:

• Chapter 10, “Operating the Apply program for SQL replication,” on page 121

Chapter 6. Subsetting data in an SQL replication environment

Some subsetting is usually involved in replication. When you register a replication source, you choose which columns and rows you want to replicate from the source table. When you create subscription sets, you choose which of the registered columns you want to replicate to each target table.

The basic subsetting methods are described in Chapter 3, "Registering tables and views as SQL replication sources," on page 35 and Chapter 4, "Subscribing to sources for SQL replication," on page 57. This chapter describes some advanced techniques that you can use to subset data. Depending on your replication requirements, you can use these techniques to subset data at the source during registration or at the target during subscription:

- If you have only one target for a source, or if all targets need exactly the same data, then it is possible to subset or manipulate at registration because you do not need to consider potentially different needs of different targets.
- If you have one source and multiple targets, and the multiple targets have different requirements regarding the data to be applied, then it might not be possible to subset at registration. In this case, you would subset data at subscription.

Do *not* use any of these techniques if you are replicating to replica target tables. The master table and replica tables in update-anywhere configurations replicate data back and forth to one another. Replica tables can have a subset of the source table columns as long as the columns that are not used are nullable. Otherwise, replica tables *must* contain the same columns as the source table so you *cannot* subset columns, add new columns, or rename columns.

This chapter contains the following sections:

- "Subsetting data during registration"
- "Subsetting data during subscription" on page 95

Subsetting data during registration

You can use advanced techniques to subset your data during registration. These techniques are especially useful if you want to capture the same subset of data once and replicate that subset to many target tables. You can choose to subset data either before or after it is captured from a registered source. The techniques in this section can be used in all replication configurations except update-anywhere or peer-to-peer replication.

Subsetting data during registration can improve replication's overall performance because it reduces the amount of data that the Capture program adds to the CD table and the amount that the Apply program reads. It also reduces storage because there are less rows in the CD table.

This section discusses the following ways that you can subset data during registration:

- "Subsetting source data using views" on page 94
- "Defining triggers on CD tables to prevent specific rows from being captured (Linux, UNIX, Windows, z/OS)" on page 94

Subsetting source data using views

When you register a source, you choose the columns that you want to make available for replication. The columns that you select are captured for replication. In some cases, after you register a source for change replication, you might want to register a view of the source.

For example, assume that the Human Resources department maintains a table that contains personnel data, including salary information. To maintain a backup database, the whole personnel table is registered and subscribed to at the backup site. However, if another target site wants to subscribe to the personnel table, you might want to hide the salary information from this second subscriber. The solution is to register a view over the personnel table, and allow access privileges on only the registered view for the second subscriber, so that the salary information is protected from access. A subscription can be created on this registered view.

You can also register views that include two or more source tables. For example, if you have a customer table and a branch table, the only way to adequately subset the customers to the target correctly might be by joining the two tables so that only the customers for a certain branch are replicated to a certain target. In this case, you must take care to avoid double-deletes.

Defining triggers on CD tables to prevent specific rows from being captured (Linux, UNIX, Windows, z/OS)

When you register a source, the Replication Center lets you select which columns you want captured, but it does *not* let you prevent certain changes in those rows from being replicated. In some replication scenarios, you might want to prevent certain changes in rows from being captured and replicated to the target tables. For example, if you want your target tables to contain all rows and you never want any rows deleted from them, you do not want to replicate deletes from the source.

To suppress certain changes from being captured, define triggers on your CD table. These triggers specify what changes the Capture program should ignore and not add a row to the CD table. You cannot create these triggers using the Replication Center, but you can manually create these triggers for an existing CD table (that is, after the source is registered). Any trigger failure that shows an SQLSTATE of 99999 is ignored by the Capture program and the row is not inserted into the CD table.

Example: Suppose that you want all source table DELETE operations to be suppressed during replication from the table SAMPLE.TABLE, where the CD table is SAMPLE.CD_TABLE. The following trigger suppresses any rows that are DELETE operations from being inserted into the CD table:

```
CREATE TRIGGER SAMPLE.CD_TABLE_TRIGGER
NO CASCADE BEFORE INSERT ON SAMPLE.CD_TABLE
REFERENCING NEW AS CD
FOR EACH ROW MODE DB2SQL
WHEN (CD.IBMSNAP_OPERATION = 'D')
SIGNAL SQLSTATE '99999' ('CD INSERT FILTER')
```

You might want to add the create trigger statement to the SQL that was generated during registration. You must run the modified SQL to complete the registration and to create the triggers on the CD tables.

These triggers execute every time the Capture program tries to insert a row in the CD table, so you need to consider if using triggers here will give you the best performance in your replication configuration. You can increase or decrease data throughput by adding triggers to CD tables. Use triggers on the CD table to suppress a significant number of changes at the source. If you plan to capture most of the changes, but want to suppress some of them from being replicated, you might want to suppress the unwanted rows during subscription.

Subsetting data during subscription

This section describes how you can use predicates to subset rows during subscription. Subsetting data during subscription can improve replication's overall performance because it reduces the amount of data that the Apply program fetches. It also reduces storage because there are less rows in the target tables.

The Apply program uses predicates to determine what data to copy during full refresh and change-capture replication. The Replication Center allows you to specify predicate values for full refresh and change-capture replication. You might want to add additional predicate information to use *only* for change-capture replication because that information is not available during full refresh. You must add this additional predicate information to the subscription set member (IBMSNAP_SUBS_MEMBR) table in the UOW_CD_PREDICATES column through SQL that you provide.

For example, suppose that you have a registered table called ALL.CUSTOMERS, and its associated CD table is called ALL.CD_CUSTOMERS. Assume that you want the subscription target to contain only a subset of ALL.CUSTOMERS where the ACCT_BALANCE column is greater than 50000, and you want to maintain historical data in the target table (that is, you do not want any data deleted from the target table). Using the Replication Center, you can create the subscription set member with a PREDICATES value of 'ACCT_BALANCE > 50000'.

You cannot use the Replication Center to prevent deletes at the target table, because the information about the type of operation is stored in the CD table and is not available at the source table or view. Therefore, you must generate the additional change-capture predicate by using an SQL statement that includes the following information[3]:

```
UPDATE ASN.IBMSNAP_SUBS_MEMBR SET UOW_CD_PREDICATES = 'IBMSNAP_OPERATION <>"D"'
   WHERE APPLY_QUAL = 'apply_qual' AND SET_NAME = 'set_name' AND
   SOURCE_OWNER = 'ALL' AND SOURCE_TABLE = 'CUSTOMERS'
```

You must set up the UOW_CD_PREDICATES column manually for any subscription-set member predicate that references any column that is not available during full refresh, including the before-image columns in the CD table, any overhead columns from the CD table, or any column from the UOW table.

By default, the Apply program does not join the UOW table and the CD table for user-copy target tables; it fetches and applies data directly from the CD table. If the predicate has to reference the UOW table, and the target table is a user copy, you must set the value of the JOIN_UOW_CD column to Y in the subscription members (IBMSNAP_SUBS_MEMBR) table. Setting this flag ensures that the Apply program joins the UOW and CD tables.

3. Depending on your scenario, you might need to add columns to the update statement to ensure that you update a single row in the subscription members (IBMSNAP_SUBS_MEMBR) table.

If you want to specify predicates that exceed 1024 bytes (the capacity of the PREDICATES column of the subscription-members (IBMSNAP_SUBS_MEMBR) table) for a row subset, you must use a source view.

If you are using complex predicate statements for a subscription set, enclose the entire expression in parentheses. For example, when using the AND and OR clauses in a predicate statement, enclose the expression as follows:

```
((TOSOURCE = 101 AND STATUS IN (202,108,109,180,21,29,32,42))
OR (SOURCE = 101))
```

Chapter 7. Manipulating data in an SQL replication environment

The data in your target tables does not need to appear exactly as it does in your source tables. You can transform or enhance your source data before it is replicated to the target tables. For example, you might want to manipulate your data in the following ways: perform data cleansing, perform data aggregation, or populate columns at the target table that do not exist at the source.

This chapter describes some advanced techniques that you can use to transform your data.

You can manipulate data either before or after it's captured from a registered source. Manipulate your data at registration instead of at subscription if you want to manipulate the data *once* and replicate transformed data to *many* target tables. Manipulate your data during subscription instead of registration if you want to capture *all* of the source data and *selectively* apply transformed data to individual targets.

In some replication scenarios, you might want to manipulate the content of the source data that is stored in the CD table. A trigger, an expression through the subscription, or a source view can all be used to get the same job done. Each method has its pros and cons. A trigger might be too costly in terms of CPU used. A view lets you set up the function once rather than in multiple subscriptions.

For example, if a particular value is missing in the source table, you might *not* want the Capture program to capture null values.

You can use triggers on your CD table to specify conditions for the Capture program to enhance the data when inserting data to the CD table. In this case, you can specify that the Capture program should insert a default value in the CD table when it encounters a null value in the source. You can use the following code to create a trigger that supplies an unambiguous default if data is missing from the source table update:

```
CREATE TRIGGER ENHANCECD
NO CASCADE BEFORE INSERT ON CD_TABLE
REFERENCING NEW AS CD
FOR EACH ROW MODE DB2SQL
WHEN (CD.COL1 IS NULL)
SET CD.COL1 ='MISSING DATA'
END
```

Instead of the trigger, you can use the COALESCE scalar function of DB2 in a registered source view or in a subscription expression. In a registered view, the coalesce function returns the first non-null value.

Partial sample using a source view:

```
CREATE VIEW SAMPLE.SRCVIEW  (columns) AS SELECT
       ... COALESCE(A.COL1, 'MISSING DATA') ...
       FROM SAMPLE.TABLE A
```

Partial sample using an expression:

```
COALESCE(CD.COL1, 'MISSING DATA')
```

The Apply program can manipulate data, either before or after it applies data to the target, in the following ways:

- "Enhancing data using stored procedures or SQL statements"
- "Mapping source and target columns that have different names"
- "Creating computed columns" on page 99

Enhancing data using stored procedures or SQL statements

When you define subscription set information, you can define run-time processing statements using SQL statements or stored procedures that you want the Apply program to run every time it processes a *specific* set. These run-time processes enable you to manipulate the data during replication. Such statements are useful for pruning CCD tables and controlling the sequence in which subscription sets are processed. You can run the run-time processing statements at the Capture control server before a subscription set is processed, or at the target server before or after a subscription set is processed. For example, you can execute SQL statements before retrieving the data, after replicating it to the target tables, or both.

Restriction for nicknames: Federated DB2 tables (using nicknames) are usually updated within a single unit of work. When you add an SQL statement to a subscription set that runs after the Apply program applies all data to the targets, you must *precede* that SQL statement with an SQL COMMIT statement in either of the following two situations:

- The SQL statement inserts into, updates, or deletes from a nickname on a server other than the server where the target tables or target nicknames for the subscription set are located.
- The SQL statement inserts into, updates, or deletes from a table local to the Apply control server, but the target nicknames for the subscription set are located on a remote server.

The extra COMMIT statement commits the Apply program's work before it processes your added SQL statement.

Stored procedures use the SQL CALL statement without parameters. The procedure name must be 18 characters or less in length (for OS/400, the maximum is 128). If the source or target table is in a non-DB2 relational database, the SQL statements are executed against the federated DB2 database. The SQL statements are never executed against a non-DB2 database. The run-time procedures of each type are executed together as a single transaction. You can also define acceptable SQLSTATEs for each statement.

Use the ASNDONE exit routine if you want to manipulate data after *each* set completes (rather than after a *specific set* completes).

Mapping source and target columns that have different names

When you are using the Replication Center to define a subscription-set member and the target table being referenced does not exist, you can rename columns at the target regardless of the target-table type. Also, you can change column attributes (data type, length, scale, precision, and whether it is nullable) where they are compatible. You cannot use the Replication Center to rename columns of existing target tables. The Replication Center will try to map the columns by name if the target table being referenced by the subscription-set member already exists. If the source and target columns do not match, you can either use the Replication

Center to map the columns from the source to the target, or you can create a view of the target table that contains a match to the source column names.

Creating computed columns

Although you cannot change the names of columns in existing target tables, you can modify the expressions of the source columns so that they map correctly to, or are compatible with, the columns in existing target tables. Using SQL expressions, you can also derive new columns from existing source columns. For aggregate target-table types, you can define new columns by using aggregate functions such as COUNT or SUM. For other types of target tables, you can define new columns using scalar functions in expressions. If the columns in source and target tables only differ by name but are otherwise compatible, you can use the Replication Center to map one column to the other.

For example, assume that you have existing source table (SRC.TABLE) and target table (TGT.TABLE):

```
CREATE TABLE SRC.TABLE (SRC_COL1 CHAR(12) NOT NULL, SRC_COL2 INTEGER,
     SRC_COL3 DATE, SRC_COL4 TIME, SRC_COL5 VARCHAR(25))
CREATE TABLE TGT.TABLE (TGT_COL1 CHAR(12) NOT NULL,
     TGT_COL2 INTEGER NOT NULL, TGT_COL3 TIMESTAMP, TGT_COL4 CHAR(5))
```

Use the following steps to map the desired target table using computed columns during subscription:

1. Use the Replication Center to map SRC_COL1 from the source table to TGT_COL1 in the target table. Since these columns are compatible, you do not have to use an expression to map one to the other.
2. Use the expression COALESCE(SRC_COL2, 0) to compute the column values and map to provide TGT_COL2. Because SRC_COL2 is nullable and TGT_COL2 is NOT NULL, you must perform this step to ensure that a NOT NULL value is provided for TGT_COL2.
3. Use the expression TIMESTAMP(CHAR(SRC_COL3) CONCAT CHAR(SRC_COL4)) to compute the column values and map to provide TGT_COL3. This column expression provides data to map to the timestamp column in the target database.
4. Use the expression SUBSTR(SRC_COL5, 1,5) to compute the column values and map to provide TGT_COL4.

Chapter 8. Customizing and running replication SQL scripts for SQL replication

To create control tables, register source tables, and create subscription sets, you must run SQL scripts that are generated by the Replication Center. You can run the SQL scripts using the Replication Center, the Task Center, or you can run them from a DB2 command line. If necessary, you can modify the SQL scripts to meet your needs.

You have the option in the Replication Center to run a generated SQL script immediately or to save the generated SQL script as a task or to a file and run the script at a later time. Even if you choose to run the SQL from the Replication Center, you might *also* want to save it as a task or to a file for future reference. For example, if you save the definitions of a large replication subscription set in an SQL file, you can rerun the definitions as necessary.

When editing the generated SQL scripts, be careful not to change the termination characters. Also, do not change the script separators if there are multiple scripts saved to a file.

You might want to customize the SQL scripts for your environment to perform the following tasks:

- Create multiple copies of the same replication action, customized for multiple servers.
- Set the size of the table spaces or databases of the CD tables.
- Define site-specific standards.
- Combine definitions together and run as a batch job.
- Defer the replication action until a specified time.
- Create libraries of SQL scripts for backup, site-specific customization, or to run stand-alone at distributed sites, such as for an occasionally-connected environment.
- Edit create table and index statements to represent database objects.
- For Informix and other non-DB2 relational databases, ensure that tables are created in the table spaces that you want.
- For Microsoft SQL Server, create control tables on an existing segment.
- Review and edit subscription-set member predicates as a way of defining multiple subscription sets at one time. You can use substitution variables in your predicates and resolve the variables with programming logic.

If you run the SQL scripts from a DB2 command line, you must connect to servers manually when you run the SQL script. The script is generated with CONNECT statements. Before you run the SQL script, you must edit the SQL statements to specify the user ID and password for the server. For example, look for a line that resembles the following example and add your information by typing over the placeholders (XXXX):

```
CONNECT TO srcdb USER XXXX USING XXXX ;
```

Procedure:

Use one of the following methods to run the files containing SQL scripts from a DB2 command line:

- Use this command if the SQL script has a semicolon (;) as a termination character:

  ```
  db2 -tvf filename
  ```

- Use this command if the SQL script has some other character as the delimiter (in this example, as in heterogeneous replication, the pound sign (#) is the termination character):

  ```
  db2 -td# -vf filename
  ```

Recommendation: Always read the administration log file before running any scripts.

Chapter 9. Operating the Capture program for SQL replication

This chapter pertains to log-based capture for DB2 databases. If you are using trigger-based capture, the triggers are created at registration, and you do not perform the operations described in this chapter.

This chapter contains the following sections:

- "Default operational parameters for the Capture program"
- "Changing operational parameters for the Capture program" on page 105
- "Starting the Capture program (Linux, UNIX, Windows, z/OS)" on page 107
- "Starting the Capture program (OS/400)" on page 115
- "Altering the behavior of a running Capture program" on page 116
- "Changing the operating parameters in the Capture parameters table" on page 117
- "Stopping the Capture program" on page 118
- "Suspending Capture (Linux, UNIX, Windows, z/OS)" on page 118
- "Resuming Capture (Linux, UNIX, Windows, z/OS)" on page 119
- "Reinitializing Capture" on page 119

Important: The Capture program does not capture any changes made by some DB2 utilities, because the utilities do not log changes in a way that is visible to the Capture program.

Default operational parameters for the Capture program

Capture has several parameters for which there are default values. The default values that are shipped with the product are shown in Table 6 and Table 7 on page 104. The default values for most operational parameters are shipped and are stored in the Capture parameters (IBMSNAP_CAPPARMS) table. Use these defaults in your replication environment and change them as necessary using one of the methods described in "Changing operational parameters for the Capture program" on page 105.

Table 6. Default settings for Capture operational parameters (Linux, UNIX, Windows, z/OS)

Operational parameter	Default value	Column name in IBMSNAP_CAPPARMS table
capture_server	DB2DBDFT[1]	not applicable
capture_schema	ASN[2]	not applicable
add_partition	n[4]	not applicable
retention_limit	10080 minutes	RETENTION_LIMIT
lag_limit	10080 minutes	LAG_LIMIT
commit_interval	30 seconds	COMMIT_INTERVAL
prune_interval	300 seconds	PRUNE_INTERVAL
trace_limit	10080 minutes	TRACE_LIMIT
monitor_limit	10080 minutes	MONITOR_LIMIT
monitor_interval	300 seconds	MONITOR_INTERVAL

Table 6. Default settings for Capture operational parameters (Linux, UNIX, Windows, z/OS) (continued)

Operational parameter	Default value	Column name in IBMSNAP_CAPPARMS table
memory_limit	32 MB	MEMORY_LIMIT
autoprune	y[3]	AUTOPRUNE
term	y[3]	TERM
autostop	n[4]	AUTOSTOP
logreuse	n[4]	LOGREUSE
logstdout	n[4]	LOGSTDOUT
sleep_interval	5 seconds	SLEEP
capture_path	Directory where Capture was started[5]	CAPTURE_PATH
startmode	warmsi[6]	STARTMODE

Notes:

1. The Capture control server is the value of the DB2DBDFT environment variable for Windows, Linux, and UNIX, if that variable is specified. There is no default value for z/OS.
2. You cannot change the default for the Capture schema. To use another Capture schema, use the **capture_schema** start-up parameter.
3. Yes
4. No
5. If Capture starts as a Windows service, its capture path is \sqllib\bin.
6. The Capture program warm starts. It switches to cold start only if this is the first time that the program is starting.

For more information about these operational parameters and their defaults, see "Starting the Capture program (Linux, UNIX, Windows, z/OS)" on page 107.

Table 7. Default settings for Capture operational parameters (OS/400)

Operational parameter	Default value	Column name in IBMSNAP_CAPPARMS table
CAPCTLLIB	ASN[1]	not applicable
JOBD	*LIBL/QZSNDPR	not applicable
JRN	*ALL	not applicable
RETAIN	10080 minutes	RETENTION_LIMIT
LAG	10080 minutes	LAG_LIMIT
FRCFRQ	30 seconds	COMMIT_INTERVAL
CLNUPITV	*IMMED [2]	not applicable
CLNUPITV	86400 seconds[2]	PRUNE_INTERVAL
CLNUPITV	*IMMED [2]	not applicable
TRCLMT	10080 minutes	TRACE_LIMIT
MONLMT	10080 minutes	MONITOR_LIMIT
MONITV	300 seconds	MONITOR_INTERVAL
MEMLMT	32 MB	MEMORY_LIMIT
WAIT	120 seconds	not applicable

Table 7. Default settings for Capture operational parameters (OS/400) (continued)

Operational parameter	Default value	Column name in IBMSNAP_CAPPARMS table
RESTART	*YES[3]	not applicable

Notes:

1. You cannot change the default for the Capture schema. To use another Capture schema, specify the CAPCTLLIB parameter when you start the Capture program. The default values for most other operational parameters are shipped and are stored in the Capture parameters (IBMSNAP_CAPPARMS) table.
2. The CLNUPITV has two sub-parameters. By default, the Capture program prunes soon after it starts running and again after every prune interval is reached (which, by default, is every 24 hours).
3. By default, the Capture program warm starts.

For more information about these operational parameters and their defaults, see Chapter 19, "System commands for SQL replication (OS/400)," on page 319

Changing operational parameters for the Capture program

You can change the default values for the operational parameters to values that you typically use in your environment. You can override these default values when you start the Capture program or modify them while the Capture program is running.

Setting new default values in the IBMSNAP_CAPPARMS table

The Capture parameters (IBMSNAP_CAPPARMS) table contains parameters that you can modify to control the operation of the Capture program. The schema name of the table is the Capture schema. After the table is created, it contains the default values that are shipped for the Capture program. If the column value in the CAPPARMS table is not set, the hard-coded default value shown in Table 6 on page 103 and Table 7 on page 104 are used. For more information about how to modify the values in this table, see "Changing the operating parameters in the Capture parameters table" on page 117.

Specifying values for parameters when you start the Capture program

You can specify values for the Capture program when you start it. The values that you set during startup control the behavior of Capture for the current session, they override the default operational parameter values and any values that might exist in the Capture parameters table. They do not update the values in the Capture parameters table. If you do not modify the Capture parameters table before you start the Capture program, and you do not specify any parameters when you start the Capture program, default values are used for the operational parameters.

Changing parameter values while the Capture program is running

While Capture is running, you can change its operational parameters temporarily. The Capture program will use the new values until you change the values again, or until you stop and restart the Capture program. You can change the Capture parameters as often as you like during the session. See "Altering the behavior of a running Capture program" on page 116 for details.

Example (Linux, UNIX, Windows): Assume that you do not want to use the default settings for the Capture commit interval for Capture schema ASNPROD.

1. Update the Capture parameters table for the ASNPROD Capture schema. Set the commit interval to 60 seconds; therefore, when you start the Capture program in the future, the commit interval will default to 60 seconds.

   ```
   update asnprod.ibmsnap_capparms set commit_interval=60;
   ```

2. Eventually you might want to do some performance tuning so you decide to try starting Capture using a lower commit interval. Instead of changing the value in the Capture parameters table, you simply start the Capture program with the commit interval parameter set to 20 seconds. While the Capture program runs using a 20-second commit interval, you monitor its performance.

   ```
   asncap capture_server=srcdb1 capture_schema=asnprod commit_interval=20
   ```

3. You decide that you want to try an even lower commit interval. Instead of stopping the Capture program, you submit a change parameters request that sets the commit interval to 15 seconds. The Capture program continues to run, only now it commits data every 15 seconds.

   ```
   asnccmd capture_server=srcdb1 capture_schema=asnprod chgparms
           commit_interval=15
   ```

   Important: The parameter that you are changing must immediately follow the **chgparms** command.

4. You can continue monitoring the performance and changing the commit interval parameter without stopping the Capture program. Eventually, when you find the commit interval that meets your needs, you can update the Capture parameters tables (as described in step 1) so that the next time you start the Capture program it uses the new value as the default commit interval.

Example (OS/400): Assume that you do not want to use the default settings for the Capture commit interval for Capture schema ASNPROD.

1. Update the Capture parameters table for the ASNPROD Capture schema. Set the commit interval to 90 seconds; therefore, when you start the Capture program in the future the commit interval will default to 90 seconds.

   ```
   CHGDPRCAPA CAPCTLLIB(ASNPROD) FRCFRQ(90)
   ```

2. Eventually you might want to do some performance tuning so you decide to try starting Capture using a lower commit interval. Instead of changing the value in the Capture parameters table, you simply start the Capture program with the commit interval parameter set to 45 seconds. As the Capture program runs using a 45-second commit interval, you monitor its performance.

   ```
   STRDPRCAP CAPCTLLIB(ASNPROD) FRCFRQ(45)
   ```

3. You decide that you want to try an even lower commit interval. Instead of stopping the Capture program, you submit a change parameters request that sets the commit interval to 30 seconds. The Capture program continues to run, only now it commits data every 30 seconds. (Note: On OS/400, you can't set the commit interval to less than 30 seconds.)

   ```
   OVRDPRCAPA CAPCTLLIB(ASNPROD) FRCFRQ(30)
   ```

4. Eventually, when you find the commit interval that meets your needs, you can update the Capture parameters tables (as described in step 1) so that the next time you start the Capture program it will use the new value as the default commit interval.

Starting the Capture program (Linux, UNIX, Windows, z/OS)

Start the Capture program to begin capturing data from the log. The Capture program captures data from DB2 databases only. If you are using trigger-based capture to capture changes from a non-DB2 relational source, the triggers are created at registration and you do not need to start the Capture program.

Important: The Capture program does not capture any changes made by DB2 utilities, because the utilities do not log changes in a way that is visible to the Capture program.

After you start the Capture program, the Capture program might not start capturing data right away. It will start capturing data only after the Apply program signals the Capture program that it has refreshed a target table fully. Then the Capture program starts capturing changes from the log for a given source table.

Tip: Look in the Capture log file (*db2instance.capture_server.capture_schema*.CAP.log on Linux, UNIX, and WINDOWS; *capture_server.capture_schema*.CAP.log on z/OS) for a message that indicates that change capture has begun. For example:

```
ASN0104I Change capture has been started for the source
table "REGRESS.TABLE1" for changes found in the log beginning
with log sequence number "0000:0275:6048".
```

Prerequisites:

Before you start the Capture program, ensure that the following prerequisites are met:

- Connections are configured to the source server and the Capture control server.
- You have the proper authorization.
- The control tables are created for the appropriate Capture schema, and registrations are defined.
- The replication programs are configured.

Procedure:

Use one of the following methods to start the Capture program on DB2 for Linux, UNIX, Windows, and z/OS:

Replication Center
Use the Start Capture window to run the Capture program identified by a Capture schema on a selected Capture control server that is in the Replication Center object tree. See the Replication Center help for details.

asncap system command
See "asncap: Starting Capture" on page 282 for command syntax and parameter descriptions.

MVS console or TSO (z/OS)
See Chapter 20, "Operating the SQL replication programs (z/OS)," on page 405 for details.

Windows Services (Windows)
See Chapter 21, "Using the Windows Service Control Manager to issue system commands for SQL replication (Windows)," on page 409 for details.

Regardless of which procedure you use to start the Capture program, you can select start-up parameters. The following sections discuss the start-up parameters and recommend when to choose one value over another for each parameter.

- "add_partition (Linux, UNIX, Windows)"
- "autoprune (Linux, UNIX, Windows, z/OS)"
- "autostop (Linux, UNIX, Windows, z/OS)" on page 109
- "capture_path (Linux, UNIX, Windows, z/OS)" on page 109
- "capture_schema (Linux, UNIX, Windows, z/OS)" on page 110
- "capture_server (Linux, UNIX, Windows, z/OS)" on page 110
- "commit_interval (Linux, UNIX, Windows, z/OS)" on page 110
- "lag_limit (Linux, UNIX, Windows, z/OS)" on page 111
- "logreuse (Linux, UNIX, Windows, z/OS)" on page 111
- "logstdout (Linux, UNIX, Windows, z/OS)" on page 111
- "memory_limit (Linux, UNIX, Windows, z/OS)" on page 112
- "monitor_interval (Linux, UNIX, Windows, z/OS)" on page 112
- "monitor_limit (Linux, UNIX, Windows, z/OS)" on page 112
- "prune_interval (Linux, UNIX, Windows, z/OS)" on page 112
- "retention_limit (Linux, UNIX, Windows, z/OS)" on page 113
- "sleep_interval (Linux, UNIX, Windows, z/OS)" on page 114
- "startmode (Linux, UNIX, Windows, z/OS)" on page 114
- "term (Linux, UNIX, Windows, z/OS)" on page 115
- "trace_limit (Linux, UNIX, Windows, z/OS)" on page 115

add_partition (Linux, UNIX, Windows)

Default: **add_partition**=n

The **add_partition** parameter specifies whether the Capture program starts reading the log file for the newly added partitions since the last time the Capture program was restarted.

Set **add_partition**=y to have the Capture program read the log files. On each new partition, when the Capture program is started in the warm start mode, Capture will read the log file starting from the first log sequence number (LSN) that DB2 used after the first database CONNECT statement is issued for the DB2 instance.

autoprune (Linux, UNIX, Windows, z/OS)

Default: **autoprune**=y

The **autoprune** parameter specifies whether or not the Capture program automatically prunes some of its control tables. By default, with **autoprune**=y, the Capture program automatically prunes the rows in the CD and UOW tables as well as Capture trace (IBMSNAP_CAPTRACE), Capture monitor (IBMSNAP_CAPMON), and signal (IBMSNAP_SIGNAL) tables. If you set **autoprune**=n, you must use the prune command to prune these tables.

If you start Capture with autopruning on, set the prune interval to optimize the pruning frequency for your replication environment. See "prune_interval (Linux, UNIX, Windows, z/OS)" on page 112 for details.

The Capture program uses the following parameters to determine which rows are old enough to prune:

- "retention_limit (Linux, UNIX, Windows, z/OS)" on page 113 for CD, UOW, and signal tables
- "monitor_limit (Linux, UNIX, Windows, z/OS)" on page 112 for monitor tables
- "trace_limit (Linux, UNIX, Windows, z/OS)" on page 115 for the Capture trace table

For more information about pruning your tables, see "Pruning your control tables" on page 212.

autostop (Linux, UNIX, Windows, z/OS)

Default: **autostop**=n

The **autostop** parameter controls whether the Capture program stays up or terminates after it reaches the end of the log.

By default (**autostop**=n) the Capture program does not terminate after retrieving the transactions.

Use the **autostop**=y option if you are replicating in a mobile or an occasionally connected environment. Autostop ensures that the Capture program retrieves all eligible transactions and stops when it reaches the end of the log. You need to start Capture again to retrieve more transactions. You might want to use the **autostop**=y option in a test environment, too.

Recommendation: In most cases you should not use **autostop**=y because it adds a lot of overhead to the administration of replication (for example, you need to keep restarting the Capture program).

capture_path (Linux, UNIX, Windows, z/OS)

The Capture path is the directory where the Capture program stores its work files and log file. By default, the Capture path is the directory where you start the program. If you start the Capture program as a Windows service, by default the Capture program starts in the \sqllib\bin directory. On the z/OS operating system, because the Capture program is a POSIX application, the default Capture path depends on how you start the program:

- If you start the Capture program from a USS command line prompt, the Capture path is the directory where you started the program.
- If you start the Capture program using a started task or through JCL, the default Capture path is the home directory of the user ID associated with the started task or job.

You can change the Capture path to specify where you want the Capture program to store its files. You can specify a path name, for example: /home/db2inst/capture_files. On z/OS operating systems, you can specify either a path name or a High Level Qualifier(HLQ), such as //CAPV8. When you use a HLQ, sequential files are created that conform to the file naming conventions for z/OS sequential data set file names. The sequential data sets are relative to the user ID that is running the progam. Otherwise these file names are similar to those stored in an explicitly named directory path, with the HLQ concatenated as the first part of the file name. For example, `sysadm.CAPV8.`*`filename`*.

capture_schema (Linux, UNIX, Windows, z/OS)

Default: **capture_schema**=ASN

The **capture_schema** parameter identifies which Capture program you want to start. By default, the Capture schema is ASN.

If you already set up another schema, you can start the Capture program by specifying that schema using the **capture_schema** parameter. See "Creating multiple sets of Capture control tables" on page 24 for instructions.

You might use multiple Capture schemas in the following situations:

Achieving application independence
Create multiple Capture schemas so that you can have one Capture program for application A and another Capture program for application B. Each Capture program uses its own control tables. If one of the Capture programs is down, only one application is affected. The other application is not affected because it is being serviced by another Capture program.

Meeting different applications' requirements
Create multiple Capture schemas if you have different applications that use the same source tables but have different data requirements. For example, a payroll application needs sensitive employee data while an internal employee registry does not. You can register the confidential information in one Capture schema, but not in the other Capture schema. Similarly, you can register a table more than once if some applications need the Capture program to behave differently. For example, perhaps some applications require that the Capture program saves updates as delete and insert pairs.

Isolating problems with registrations
If you have a problem with one registration, you can create another Capture schema and move the working registrations to it. That way you can debug the problem registration in the original schema and run the unaffected registrations using the other schema.

capture_server (Linux, UNIX, Windows, z/OS)

Default (Linux, UNIX, Windows): **capture_server**=value of DB2DBDFT environment variable, if it is set

Default (z/OS): **capture_server**=None

The **capture_server** parameter specifies the Capture control server. The capture control tables (such as the register table) contain the registration information for the source tables and are located at the capture control server if you are running Capture on Linux, UNIX, and Windows operating systems. If you are running Capture on z/OS, the capture control tables are located at the DB2 subsystem name. Because the Capture program reads the DB2 log, the Capture program must run at the same server as the source database.

commit_interval (Linux, UNIX, Windows, z/OS)

Default: **commit_interval**=30

The **commit_interval** parameter specifies how often, in seconds, the Capture program commits data to the Capture control tables, including the UOW and CD tables. By default, the Capture program waits 30 seconds before committing data to the CD and UOW tables. Locks are held on the tables updated within the

commit interval. Higher values for the **commit_interval** parameter reduce CPU usage for the Capture program but also might increase the latency for frequently running subscription sets because the Apply program can fetch only committed data.

lag_limit (Linux, UNIX, Windows, z/OS)

Default: **lag_limit**=10 080

The **lag_limit** parameter represents the number of minutes that the Capture program can lag in processing records from the DB2 log.

By default, if log records are older than 10 080 minutes (seven days), the Capture program will not start unless you specify a value for the **startmode** parameter that allows the Capture program to switch to a cold start.

If the Capture program will not start because the lag limit is reached, you should determine why the Capture program is behind in reading the log. If you are in a test environment, where you have no practical use for the lag limit parameter, you might want to set the lag limit higher and try starting the Capture program again. Alternatively, if you have very little data in the source table in your test environment, you might want to cold start the Capture program and fully refresh the data in all the target tables.

logreuse (Linux, UNIX, Windows, z/OS)

Default: **logreuse**=n

The Capture program stores operational information in a log file.

On Linux, UNIX, and Windows operating systems, the name of the log file is *db2instance.capture_server.capture_schema*.CAP.log. For example, DB2INST.SRCDB1.ASN.CAP.log.

On the z/OS operating system, the file name is similar except that it does not contain a DB2 instance name. For example, SRCDB1.ASN.CAP.log. This file is stored in the directory that is specified by the **capture_path** parameter. If the **capture_path** parameter is specified as a High Level Qualifier (HLQ), the file naming conventions of z/OS sequential data set files apply; therefore, the **capture_schema** name that is used to build the log file name is truncated to the first 8 characters of the name.

By default (**logreuse**=n), the Capture program appends messages to the log file, even after the Capture program is restarted. Keep the default if you want the history of the messages. In the following situations you might want the Capture program to delete the log and re-create it when it restarts (**logreuse**=y):

- The log is getting large and you want to clean out the log.
- You don't need the history that is stored in the log.
- You want to save space.

logstdout (Linux, UNIX, Windows, z/OS)

Default: **logstdout**=n

The **logstdout** parameter is available only if you use the **asncap** command, it is not available in the Replication Center.

By default, the Capture program sends some warning and informational messages only to the log file. You might choose to send such messages to standard output (**logstdout**=y) if you are troubleshooting or if you are monitoring how your Capture program is operating in a test environment.

memory_limit (Linux, UNIX, Windows, z/OS)

Default: **memory_limit**=32

The **memory_limit** parameter specifies the amount of memory, in megabytes, that the Capture program can use.

By default, the Capture program uses 32 megabytes of memory to store transaction information before it spills to a file located in the **capture_path** directory. You can modify the memory limit based on your performance needs. Setting the memory limit higher can improve the performance of Capture but decreases the memory available for other uses on your system. Setting the memory limit lower frees memory for other uses. If you set the memory limit too low and the Capture program spills to a file, you will use more space on your system and the I/O will slow down your system.

You can monitor the memory limit by using the Replication Alert Monitor. You can also use the data in the CAPMON table to determine the number of source system transactions spilled to disk due to memory restrictions. Sum the values in the TRANS_SPILLED column of the CAPMON table.

monitor_interval (Linux, UNIX, Windows, z/OS)

Default: **monitor_interval**=300

The **monitor_interval** parameter specifies how often the Capture program writes information to the Capture monitor (IBMSNAP_CAPMON) table.

By default, the Capture program inserts rows into the Capture monitor table every 300 seconds (5 minutes). This operational parameter works in conjunction with the commit interval. If you are interested in monitoring data at a granular level, use a monitor interval that is closer to the commit interval.

monitor_limit (Linux, UNIX, Windows, z/OS)

Default: **monitor_limit**=10080

The **monitor_limit** parameter specifies how old the rows must be in the monitor table before they can be pruned.

By default, rows in the Capture monitor (IBMSNAP_CAPMON) table that are older than 10 080 minutes (seven days) are pruned. The IBMSNAP_CAPMON table contains operational statistics for the Capture program. Use the default monitor limit if you need less than one week of statistics. If you monitor the statistics frequently, you probably do not need to keep one week of statistics and can set a lower monitor limit so that the Capture monitor table is pruned more frequently and older statistics are removed. If you want to use the statistics for historical analysis and you need more than one week of statistics, increase the monitor limit.

prune_interval (Linux, UNIX, Windows, z/OS)

Default: **prune_interval**=300

The **prune_interval** parameter specifies how often the Capture program tries to prune old rows from some of its control tables. This parameter is valid *only* if **autoprune**=y.

By default, the Capture program prunes the CD and UOW tables every 300 seconds (five minutes). If the tables are not pruned often enough, the table space that they are in can run out of space, which forces the Capture program to stop. If they are pruned too often or during peak times, the pruning can interfere with application programs running on the same system. You can set the optimal pruning frequency for your replication environment. Performance will generally be best when the tables are kept small.

Before you lower the prune interval, ensure that data is being applied frequently so that pruning can occur. If the Apply program is not applying data frequently, it is useless to set the prune interval lower because the Apply program must replicate the data to all targets before the CD and UOW tables can be pruned.

The prune interval determines how often the Capture program *tries* to prune the tables. It works in conjunction with the following parameters, which determine *when* data is old enough to prune: **trace_limit**, **monitor_limit**, **retention_limit**. For example, if the **prune_interval** is 300 seconds and the **trace_limit** is 10080 seconds, the Capture program will try to prune every 300 seconds. If it finds any rows in the trace table that are older than 10080 minutes (7 days), it will prune them.

For more information about pruning your tables, see "Pruning your control tables" on page 212.

retention_limit (Linux, UNIX, Windows, z/OS)

Default: **retention_limit**=10 080

The **retention_limit** parameter determines how long old data remains in the CD, UOW, and signal (IBMSNAP_SIGNAL) tables before becoming eligible for retention limit pruning.

If the normal pruning process is inhibited due to deactivated or infrequently run subscription sets, data remains in the CD and UOW tables for long periods of time. If this data becomes older than the current DB2 timestamp minus the retention limit value, the retention limit pruning process deletes this data from the tables. If you run your subscription sets very infrequently or stop your Apply programs, your CD and UOW tables can grow very large and become eligible for retention limit pruning.

Your target tables must be refreshed to synchronize them with the source if *any* of the rows that are pruned are candidates for replication but for some reason they *were not yet applied* to the target table. You can avoid a full refresh from happening by using higher retention limits; however, your CD and UOW tables will grow and use space on your system.

If you are doing update-anywhere replication, retention limit pruning ensures that rejected transactions are deleted. Rejected transactions result if you use conflict detection with replica target tables and conflicting transactions are detected. The rows in the CD and UOW tables that pertain to those rejected transactions are not replicated and they are pruned when the retention limit is reached. A full refresh is not required if *all* the old rows that were deleted pertained to rejected transactions.

Retention pruning also ensures that signal information that is no longer required is deleted from the signal (IBMSNAP_SIGNAL) table.

For details about pruning your control tables, see "Pruning your control tables" on page 212.

sleep_interval (Linux, UNIX, Windows, z/OS)

Default: **sleep_interval**=5

The sleep interval is the number of seconds that the Capture program waits before it reads the log again after it reaches the end of the log and the buffer is empty. For data sharing on the z/OS operating system, the sleep interval represents the number of seconds that the Capture program sleeps after the buffer returns less than half full.

By default, the Capture program sleeps 5 seconds. Change the sleep interval if you want to reduce the overhead of the Capture program reading the log. A smaller sleep interval means there is less chance of delay. A larger sleep interval gives you potential CPU savings in a sparsely updated system.

startmode (Linux, UNIX, Windows, z/OS)

Default: **startmode**=warmsi

You can start Capture using one of the following start modes:

Warmsi (warm start, switch initially to cold start)
The Capture program warm starts; except if this is the first time you're starting the Capture program then it switches to cold start. Use this start mode if you want to ensure that cold starts only happen when you start the Capture program initially.

Warmns (warm start, never switch to cold start)
The Capture program warm starts. If it can't warm start, it does not switch to cold start. When you use **warmns** in your day-to-day replication environment, you have an opportunity to repair any problems (such as unavailable databases or table spaces) that are preventing a warm start from occurring. Use this start mode to prevent a cold start from occurring unexpectedly. When the Capture program warm starts, it resumes processing where it ended. If errors occur after the Capture program started, the Capture program terminates and leaves all tables intact.

Tip: You cannot use **warmns** to start the Capture program for the first time because there is no warm start information when you initially start the Capture program. Use the **cold** startmode the first time you start the Capture program, then use the **warmns** startmode. If you do not want to switch startmodes, you can use **warmsi** instead.

Warmsa (warm start, always switch to cold as necessary)
If warm start information is available, the Capture program resumes processing where it ended in its previous run. If the Capture program cannot warm start, it switches to a cold start. Usually you do not want to switch to a cold start because that requires all your target tables to be refreshed.

Cold During cold start, the Capture program deletes all rows in its CD tables and UOW table during initialization. All subscription sets to these replication sources are fully refreshed during the next Apply processing

cycle (that is, all data is copied from the source tables to the target tables). If the Capture program tries to cold start but you disabled full refresh, the Capture program will start, but the Apply program will fail and will issue an error message.

You rarely want to explicitly request that the Capture program performs a cold start. Cold start is necessary only the first time the Capture program starts, and **warmsi** is the recommended start mode.

Important: Do *not* cold start the Capture program if you want to maintain accurate histories of change data. A gap might occur if the Apply program cannot replicate changes before the Capture program shuts down. Also, because you want to avoid cold starts, do *not* put cold start as the default for STARTMODE in the Capture parameters (IBMSNAP_CAPPARMS) table.

term (Linux, UNIX, Windows, z/OS)

Default: **term**=y

The **term** parameter determines how the status of DB2 affects the operation of the Capture program.

By default, the Capture program terminates if DB2 terminates.

Use **term**=n if you want the Capture program to wait for DB2 to start if DB2 is not active. If DB2 quiesces, Capture does not terminate; it remains active but it does not use the database.

trace_limit (Linux, UNIX, Windows, z/OS)

Default: **trace_limit**=10 080

The **trace_limit** specifies how old the rows must be in the Capture trace (IBMSNAP_CAPTRACE) table before they are pruned.

When Capture prunes, by default, the rows in the Capture trace (IBMSNAP_CAPTRACE) table are eligible to be pruned every 10 080 minutes (seven days). The CAPTRACE table contains the audit trail information for the Capture program. Everything that Capture does is recorded in this table; therefore this table can grow very quickly if the Capture program is very active. Modify the trace limit depending on your need for audit information.

Starting the Capture program (OS/400)

Start the Capture program to begin capturing data from the journal.

After you start the Capture program, the Capture program might not start capturing data right away. It will start capturing data only after the Apply program signals the Capture program to start capturing changes from the log for a given source table.

Prerequisites:

Before you start the Capture program, follow the instructions in Chapter 2, "Configuring servers for SQL replication," on page 15 to ensure that the following prerequisites are met:

- You have the proper authorization.

- The control tables are created for the appropriate Capture schema, and registrations are defined.
- The replication programs are configured if the Capture program is reading a remote journal.

Procedure:

Use one of the following methods to start the Capture program on OS/400:

Replication Center
Use the Start Capture window to run the Capture program identified by a Capture schema on a selected Capture control server that is in the Replication Center object tree. See the Replication Center help for details.

STRDPRCAP system command (OS/400)
See "STRDPRCAP: Starting Capture (OS/400)" on page 393 for command syntax and parameter descriptions.

Altering the behavior of a running Capture program

While the Capture program is running, you can alter its behavior by overriding the values of one or more operational parameters. The changes are not written to the Capture parameters (IBMSNAP_CAPPARMS) table. The Capture program uses the new values until you stop the Capture program or until you supply new values.

On Linux, UNIX, Windows, and z/OS, you can change the following Capture parameters while the Capture program is running:

- Autoprune
- Autostop
- Commit_interval
- Lag_limit
- Logreuse
- Logstdout
- Memory_limit
- Monitor_interval
- Monitor_limit
- Prune_interval
- Retention_limit
- Sleep_interval
- Term
- Trace_limit

On OS/400, you can override the values for the following operational parameters for a given Capture schema:

- CLNUPITV
- FRCFRQ
- MEMLMT
- MONLMT
- MONITV
- PRUNE
- RETAIN

- TRCLMT

When you change the values, the effects might not be immediate for all parameters.

Prerequisites:

The Capture program with the specific Capture schema must be started.

Procedure:

Use one of the following methods to change the current values for the parameters for the current session:

Replication Center
In the Replication Center, use the Change Parameters for Running Capture Program window while the Capture program is running. This method allows you to see the current values of the parameters used by the running Capture program before changing them. See the Replication Center help for details.

chgparms system parameter (Linux, UNIX, Windows, z/OS)
This method does not show the current values of the parameters. The specified new values are sent to the running Capture instance. See "asnccmd: Operating Capture" on page 288.

OVRDPRCAPA system command (OS/400)
See "OVRDPRCAPA: Overriding DPR capture attributes (OS/400)" on page 375.

Changing the operating parameters in the Capture parameters table

The Capture parameters (IBMSNAP_CAPPARMS) table contains the operational parameters for the Capture program. When you start the Capture program, it uses values from this table for its default operational behavior, unless you provide new values using the start-up parameters.

Only one row is allowed in the Capture parameters table. If you want to change one or more of the default values, you can update columns instead of inserting rows. If you delete the row, the Capture program will still start using the shipped defaults, unless those defaults are overridden by the start-up parameters.

The Capture program reads this table only during startup; therefore, you should stop and start the Capture program if you want the Capture program to run with the new settings. Changing the Capture parameters table while the Capture program is running and reinitializing the Capture program will not change the operation of the Capture program. See Chapter 24, "Table structures for SQL replication," on page 421 for descriptions of the columns in this table.

Procedure:

Use one of the following methods to change the global operating parameters that are used by the Capture program and are stored in the Capture parameters (IBMSNAP_CAPPARMS) table:

Replication Center
In the Replication Center, use the Manage Capture Parameters window to view or change any of the values in the Capture parameters table. See the Replication Center help for details.

CHGDPRCAPA system command (OS/400)
See "CHGDPRCAPA: Changing DPR Capture attributes (OS/400)" on page 355.

The parameter changes take effect only after you stop and start the Capture program.

Stopping the Capture program

You can stop the Capture program for a particular Capture schema. When you stop the Capture program, it no longer captures data from the source.

OS/400: If you choose to reorganize the UOW table and all the CD tables that were open at the time that the Capture program stopped, the Capture program needs time to shut down (it does not shut down immediately).

Prerequisites:

The Capture program with the specific Capture schema must be started.

Procedure:

Use one of the following methods to stop the Capture program for the specific Capture schema:

Replication Center
In the Replication Center, use the Stop Capture window to stop the running Capture program for the selected Capture schema. See the Replication Center help for details.

asnccmd stop system command (Linux, UNIX, Windows, z/OS)
See "asnccmd: Operating Capture" on page 288.

ENDDPRCAP system command (OS/400)
See "ENDDPRCAP: Stopping Capture (OS/400)" on page 364.

If you stop or suspend the Capture program during pruning, pruning is also suspended. When you resume or restart the Capture program, pruning resumes based on the **autoprune** parameter.

You do not need to stop the Capture program to drop a registration. Always deactivate the registration before you drop it. For details, see "Stop capturing changes for registered objects" on page 177.

Suspending Capture (Linux, UNIX, Windows, z/OS)

Suspend the Capture program to relinquish operating system resources to operational transactions during peak periods without destroying the Capture program environment. Suspend the Capture program instead of stopping it if you do not want the Capture program to shut down after it finishes work in progress. When you tell the Capture to resume, you do not require the overhead of Capture starting again.

Important: Do not suspend the Capture program before you remove a replication source. Instead, deactivate then remove the replication source.

Prerequisites:

The Capture program with the specific Capture schema must be started.

Procedure:

Use one of the following methods to suspend the Capture program while it is running:

Replication Center
In the Replication Center, use the Suspend Capture window to suspend the Capture program. See the Replication Center help for details.

asnccmd suspend system command
See Chapter 18, "System commands for SQL replication (Linux, UNIX, Windows, z/OS)," on page 271.

If you stop or suspend the Capture program during pruning, pruning is also suspended. When you resume or restart the Capture program, pruning resumes based on the **autoprune** parameter.

Resuming Capture (Linux, UNIX, Windows, z/OS)

You must resume a suspended Capture program if you want it to start capturing data again.

Prerequisites:

The Capture program with the specific Capture schema must be suspended.

Procedure:

Use one of the following methods to resume the Capture program if it is suspended:

Replication Center
In the Replication Center, use the Resume Capture window to resume a suspended Capture program. See the Replication Center help for details.

asnccmd resume system command
See Chapter 18, "System commands for SQL replication (Linux, UNIX, Windows, z/OS)," on page 271.

If you stop or suspend the Capture program during pruning, pruning is also suspended. When you resume or restart the Capture program, pruning resumes based on the **autoprune** parameter.

Reinitializing Capture

Reinitialize the Capture program if you change any attributes of existing registered objects while the Capture program is running. For example, if you change the CONFLICT_LEVEL, CHGONLY, RECAPTURE, CHG_UPD_TO_DEL_INS values in the register (IBMSNAP_REGISTER) table.

For Capture on OS/400, reinitialize is also needed to start capturing data for a journal that was not being captured previously.

Prerequisites:

The Capture program with the specific Capture schema must be started.

Procedure:

Use one of the following methods to reinitialize the Capture program while its running:

Replication Center
In the Replication Center, use the Reinitialize Capture window to reinitialize the Capture program. See the Replication Center help for details.

asnccmd reinit system command
See Chapter 18, "System commands for SQL replication (Linux, UNIX, Windows, z/OS)," on page 271.

INZDPRCAP system command
See "INZDPRCAP: Reinitializing DPR Capture (OS/400)" on page 374.

Related tasks:

- Chapter 20, "Operating the SQL replication programs (z/OS)," on page 405
- Chapter 21, "Using the Windows Service Control Manager to issue system commands for SQL replication (Windows)," on page 409

Related reference:

- "asnscrt: Creating a DB2 replication service to start the replication programs" on page 302
- "asnccmd: Operating Capture" on page 288
- "asncap: Starting Capture" on page 282
- "ENDDPRCAP: Stopping Capture (OS/400)" on page 364
- "STRDPRCAP: Starting Capture (OS/400)" on page 393

Chapter 10. Operating the Apply program for SQL replication

This chapter describes how to start and stop the Apply program. It also describes how to use the ASNDONE and ASNLOAD exit routines.

This chapter contains the following sections:

- "Default operational parameters for the Apply program"
- "Changing operational parameters for the Apply program" on page 122
- "Starting the Apply program (Linux, UNIX, Windows, z/OS)" on page 123
- "Starting the Apply program (OS/400)" on page 131
- "Changing the operating parameters in the Apply parameters table (Linux, UNIX, Windows, z/OS)" on page 132
- "Stopping the Apply program" on page 133
- "Modifying the ASNDONE exit routine (Linux, UNIX, Windows, z/OS)" on page 133
- "Modifying the ASNDONE exit routine (OS/400)" on page 134
- "Refreshing target tables using the ASNLOAD exit routine" on page 135

Default operational parameters for the Apply program

Apply has several parameters for which there are default values. The default values that are shipped with the product are shown in Table 8 for Linux, UNIX, Windows and z/OS, and Table 9 on page 132 for OS/400. The default values for most operational parameters are shipped and are stored in the Apply parameters (IBMSNAP_APPPARMS) table. Use these defaults in your replication environment and change them as necessary using one of the methods described in "Changing operational parameters for the Apply program" on page 122.

Table 8. Default settings for Apply operational parameters (Linux, UNIX, Windows, z/OS)

Operational parameter	Default value	Column name in IBMSNAP_APPPARMS table
apply_qual	No default	APPLY_QUAL
apply_path	Directory where Apply was started[1]	APPLY_PATH
control_server	DB2DBDFT[2]	not applicable
copyonce	n[3]	COPYONCE
db2_subsystem	No default[4]	not applicable
delay	6 seconds	DELAY
errwait	300 seconds	ERRWAIT
inamsg	y[5]	INAMSG
loadxit	n[3]	LOADXIT
logreuse	n[3]	LOGREUSE
logstdout	n[3]	LOGSTDOUT
notify	n[3]	NOTIFY
opt4one	n[3]	OPT4ONE
pwdfile	asnpwd.aut	not applicable

Table 8. Default settings for Apply operational parameters (Linux, UNIX, Windows, z/OS) (continued)

Operational parameter	Default value	Column name in IBMSNAP_APPPARMS table
spillfile	disk[6]	SPILLFILE
sleep	y[5]	SLEEP
sqlerrcontinue	n[3]	SQLERRCONTINUE
term	y[5]	TERM
trlreuse	n[3]	TRLREUSE

Notes:

1. If Apply starts as a Windows service, its path is sqllib\bin
2. The Apply control server is the value of the DB2DBDFT environment variable, if specified. For Linux, UNIX, and Windows operating systems only.
3. no
4. The DB2 subsystem name can be a maximum of four characters. This parameter is required. The DB2 subsystem name is only applicable to z/OS operating systems.
5. yes
6. On z/OS operating systems, the default is MEM.

For more information about these operational parameters and their defaults, see "Starting the Apply program (Linux, UNIX, Windows, z/OS)" on page 123.

Changing operational parameters for the Apply program

You can change the default values for the operational parameters to values that you typically use in your environment. You can also override these default values when you start the Apply program.

Setting new default values in the IBMSNAP_APPPARMS table

The Apply parameters (IBMSNAP_APPPARMS) table contains parameters that you can modify to control the operation of the Apply program. After the table is created, it contains the default values that are shipped for the Apply program. If the column value in the APPPARMS table is not set, the hard-coded default value shown in Table 8 on page 121 for Linux, UNIX, Windows, and z/OS, and Table 9 on page 132 for OS/400 are used.

Specifying values for parameters when you start the Apply program

You can specify values for the Apply program when you start it. The values that you set during startup control the behavior of Apply for the current session, they override the default operational parameter values and any values that might exist in the Apply parameters table. They do not update the values in the Apply parameters table. If you do not modify the Apply parameters table before you start the Apply program, and you do not specify any of the optional parameters when you start the Apply program, default values are used for the operational parameters.

Example (Linux, UNIX, Windows): Assume that you do not want to use the default settings for **errwait** for the Apply qualifier ASNPROD. Update the Apply parameters table for the ASNPROD Apply qualifier. Set the **errwait** interval to 600 seconds.

```
update asn.ibmsnap_appparms set errwait=600 where apply_qual='ASNPROD';
```

Starting the Apply program (Linux, UNIX, Windows, z/OS)

You can start an instance of the Apply program to begin applying data to your targets.

After you start the Apply program, it runs continuously (unless you used the **copyonce** start-up parameter) until one of the following events occurs:

- You stop the Apply program using the Replication Center or a command.
- The Apply program cannot connect to the Apply control server.
- The Apply program cannot allocate memory for processing.

See "Checking for current status of replication programs (Linux, UNIX, Windows, z/OS)" on page 165 to learn how you can query the status of the Apply program.

Prerequisites:

Before you start the Apply program, follow the instructions in Chapter 2, "Configuring servers for SQL replication," on page 15 to ensure that your system is set up correctly:

- Connections are configured to all necessary replication servers.
- You have the proper authorization.
- The control tables that contain the source and control data for the desired Apply qualifier are created.
- The replication programs are configured.
- On z/OS, you manually bound the Apply program to all necessary servers.
- A password file exists for end-user authentication for remote servers running on Linux, UNIX, and Windows.

Also, make sure that the following conditions are met:

- At least one active subscription set exists for the Apply qualifier and that the subscription set contains one or more of the following items:
  - Subscription-set member
  - SQL statement
  - Procedure
- All condensed target tables must have a target key, which is a set of unique columns, either a primary key or unique index, that the Apply program uses to track which changes it replicates during each Apply cycle. (Non-condensed CCD tables do not have primary keys or unique indexes.)

Procedure:

Use one of the following methods to start the Apply program:

Replication Center
Use the Start Apply window. See the Replication Center help for details.

asnapply system command (Linux, UNIX, Windows, z/OS)
See "asnapply: Starting Apply" on page 276 for details.

Windows Services (Windows)
See Chapter 21, "Using the Windows Service Control Manager to issue system commands for SQL replication (Windows)," on page 409 for details.

MVS console or TSO (z/OS)
See Chapter 20, "Operating the SQL replication programs (z/OS)," on page 405 for details.

Regardless of which procedure you use to start the Apply program, you must set up the startup parameters. The following sections discuss the start-up parameters and recommend when to choose one value over another for each parameter.

- "apply_path (Linux, UNIX, Windows, z/OS)"
- "apply_qual (Linux, UNIX, Windows, z/OS)" on page 125
- "control_server (Linux, UNIX, Windows, z/OS)" on page 125
- "copyonce (Linux, UNIX, Windows, z/OS)" on page 125
- "db2_subsystem (z/OS)" on page 126
- "delay (Linux, UNIX, Windows, z/OS)" on page 126
- "errwait (Linux, UNIX, Windows, z/OS)" on page 126
- "inamsg (Linux, UNIX, Windows, z/OS)" on page 127
- "loadxit (Linux, UNIX, Windows, z/OS)" on page 127
- "logreuse (Linux, UNIX, Windows, z/OS)" on page 127
- "logstdout (Linux, UNIX, Windows, z/OS)" on page 127
- "notify (Linux, UNIX, Windows, z/OS)" on page 128
- "opt4one (Linux, UNIX, Windows, z/OS)" on page 128
- "pwdfile (Linux, UNIX, Windows)" on page 128
- "sleep (Linux, UNIX, Windows, z/OS)" on page 128
- "spillfile (Linux, UNIX, Windows, z/OS)" on page 129
- "sqlerrcontinue (Linux, UNIX, Windows, z/OS)" on page 129
- "term (Linux, UNIX, Windows, z/OS)" on page 130
- "trlreuse (Linux, UNIX, Windows, z/OS)" on page 130

apply_path (Linux, UNIX, Windows, z/OS)

Default (Linux, UNIX, Windows, z/OS): **apply_path**=*current_directory*

Default (service on Windows): **apply_path**=sqllib\bin

The Apply path is the directory where the Apply program stores its log and work files. By default, the Apply path is the directory where you start the program. You can change the Apply path to store the log and work files elsewhere (for example /home/db2inst/apply_files on an AIX system). Keep track of what directory you choose because you might need to go to this directory to access the Apply log file. On z/OS operating systems, see the SASNSAMP(ASNSTRA) job for information on how you can change the Apply path.

Important: Make sure that the directory that you choose has enough space for the temporary files used by the Apply program. For details, see "Planning space requirements for spill files for the Apply program" on page 9.

Starting instances of Apply on one Windows system: When you start the Apply program using either the Replication Center or the **asnapply** command, you *must* specify the Apply path *if* you have two or more Apply qualifiers that are identical except for their capitalization. File names on Windows systems are not case-sensitive. For example, assume that you have three Apply qualifiers:

APPLYQUAL1, ApplyQual1, applyqual1. Each of these Apply instances must be started with a different **apply_path** to prevent file name conflicts of the log files for each instance of the Apply program.

apply_qual (Linux, UNIX, Windows, z/OS)

You must specify the Apply qualifier for the subscription sets that you want to process. (You defined the Apply qualifier when you created your subscription set.) You can specify only one Apply qualifier per start command.

Important: The Apply qualifier is case-sensitive and the value that you enter must match the value of the APPLY_QUAL column in the subscription sets (IBMSNAP_SUBS_SET) table.

If you have more than one Apply qualifier defined, you can start another instance of the Apply program. Each instance of the Apply program that you start will process different subscription sets that are represented in the same Apply control server. For example, assume that you have two subscription sets defined and each set has a unique Apply qualifier: APPLY1 and APPLY2. You can start two instances of the Apply program (one for each Apply qualifier), and each instance uses the control tables on the Apply control server called CNTRLSVR. Each instance of Apply processes its own subscription sets independently, providing better performance than if a single instance of Apply processes all the sets.

control_server (Linux, UNIX, Windows, z/OS)

Default (Linux, UNIX, Windows): The value of the DB2DBDFT environment variable, if available

Default (z/OS): None

The Apply control server is the server on which the subscription definitions and the Apply control tables reside. Specify only one control server per Apply qualifier. If you do not specify a value, the Apply program will start on the default server. The default depends on your operating system.

If the Apply program cannot connect to the control server, the Apply program terminates. If it can't connect to other servers, it does not terminate. In this case it issues an error message and continues processing.

copyonce (Linux, UNIX, Windows, z/OS)

Default: **copyonce**=n

The **copyonce** parameter determines the copy cycle for the Apply program.

When you start the Apply program using **copyonce**=y, it processes each eligible subscription set only once, and then it terminates. In this case, a subscription set is eligible to be processed if one of the following conditions is met:

- The subscription set uses relative timing, the time has elapsed, and the subscription set is activated.
- The subscription set uses event-based timing, it is activated, and the event has occurred but the Apply program hasn't processed the subscription set yet.

Typically you want to start the Apply program using **copyonce**=n because you want the Apply program to continue running and processing eligible subscriptions.

If you are running the Apply program from a dial-in environment that is occasionally connected to the network, use **copyonce**=y instead of **copyonce**=n. You might also want to use **copyonce**=y if you are running the Apply program in a test environment.

Tip: Use **sleep**=n instead of **copyonce**=y if you want the Apply program to process each subscription set multiple times, as long as the set is eligible and data is available for replication. **Copyonce**=y processes each set only once even if there is more data to replicate.

db2_subsystem (z/OS)

The **db2_subsystem** parameter specifies the name of the DB2 subsystem, if you are running Apply on z/OS. The DB2 subsystem name that you enter can be a maximum of four characters. There is no default for this parameter. This parameter is required.

delay (Linux, UNIX, Windows, z/OS)

Default: **delay**=6 seconds

The **delay** parameter sets an amount of time in seconds that the Apply program waits at the end of the Apply cycle.

By default, during continuous replication (that is, when your subscription set uses **sleep**=0 minutes), the Apply program waits 6 seconds after a subscription set is processed successfully before retrying the subscription set. Use a non-zero delay value to save CPU cycles when there is no database activity to be replicated. Use a lower delay value for low latency.

Note: The **delay** parameter is ignored if **copyonce** is specified.

errwait (Linux, UNIX, Windows, z/OS)

Default: **errwait**=300 seconds (5 minutes)

The **errwait** parameter specifies the number of seconds that the Apply program waits before retrying a subscription set after a subscription cycle failed

By default, the Apply program waits 300 seconds before it retries a subscription set after a subscription cycle failed. You might want to use a smaller value in a test environment. The minimum value is 1 second. In a production environment, consider the trade-offs before you change the default for this parameter:

- If you use a smaller value, you might waste CPU cycles if the Apply program keeps retrying hard errors. For example, you will use CPU cycles unnecessarily if the Apply program keeps retrying to process a subscription set when there is a problem with a target table. You might get an undesirably large number of messages in the log file and, if the Apply program runs on z/OS, on the operator console.
- If you use a larger value, you might increase latency if the Apply program must wait to retry transient error conditions. For example, you will increase latency if you use a larger value for the **errwait** parameter because the Apply program waits unnecessarily after it encounters a network error that might be corrected quickly.

Note: The **errwait** parameter is ignored if **copyonce** is specified.

inamsg (Linux, UNIX, Windows, z/OS)

Default: **inamsg**=y

The **inamsg** parameter specifies whether or not the Apply program issues a message when it becomes inactive.

By default, the Apply program issues a message when it becomes inactive. You might not want the Apply program to issue a message when it becomes inactive because the messages will take up a lot of space in the Apply log file, especially if the Apply program is not waiting long between processing subscription sets. To turn off these messages, use **inamsg**=n.

loadxit (Linux, UNIX, Windows, z/OS)

Default: **loadxit**=n

The **loadxit** parameter specifies whether or not the Apply program should refresh target tables using the ASNLOAD exit routine.

By default, the Apply program does not use the ASNLOAD exit routine to refresh target tables (**loadxit**=n). Use **loadxit**=y if you want the Apply program to invoke the ASNLOAD exit routine to refresh target tables. Consider using the ASNLOAD exit if there is a large amount of data to be copied to the target tables during a full refresh. For information about when you might want the Apply program to use ASNLOAD and for instructions on how to use ASNLOAD, see "Refreshing target tables using the ASNLOAD exit routine" on page 135.

logreuse (Linux, UNIX, Windows, z/OS)

Default: **logreuse**=n

The Apply program stores operational information in a log file. On Linux, UNIX, and Windows, the name of the log file is *db2instance.control_server.apply_qualifier*.APP.log. On the z/OS operating system, the file name is similar, except it does not contain the DB2 instance name.

The parameter specifies whether to append to the log file or to overwrite it.

By default, the Apply program appends messages to the log file (**logreuse**=n) each time that you start the Apply program. Keep the default if you want the history of the messages that are issued by the Apply program. In the following situations you might want to use **logreuse**=y, where the Apply program deletes the log and re-creates the log when it starts:

- The log is getting large, and you want to clean out the log to save space.
- You don't need the history that is stored in the log.

logstdout (Linux, UNIX, Windows, z/OS)

Default: **logstdout**=n

The **logstdout** parameter is available only if you use the **asnapply** command; **logstdout** is not available in the Replication Center.

The **logstdout** parameter specifies whether the Apply program sends completion messages (ASN10251) to both the log file and to standard output.

By default, the Apply program does not send completion messages to standard output (STDOUT). If you specify **logstdout**=y, the Apply program will send completion messages to both the log file and to standard output (STDOUT). You might choose to send messages to standard output if you are troubleshooting or monitoring how your Apply program is operating.

notify (Linux, UNIX, Windows, z/OS)

Default: **notify**=n

The **notify** parameter specifies whether the Apply program notifies the ASNDONE exit routine after it processes a subscription.

By default, the Apply program does not notify the ASNDONE exit routine after subscription processing completes. If you specify **notify**=y, after the Apply program completes a subscription cycle it invokes ASNDONE to perform additional processing, such as examining the Apply control tables or sending e-mail messages. For more information about ASNDONE, see "Modifying the ASNDONE exit routine (Linux, UNIX, Windows, z/OS)" on page 133.

opt4one (Linux, UNIX, Windows, z/OS)

Default: **opt4one**=n

The **opt4one** parameter specifies whether or not the Apply program processing is optimized for one subscription set.

Note: The **opt4one** parameter is ignored if **copyonce** is specified.

By default, the Apply program is optimized for many subscription sets. The Apply program reads the information from the replication control tables at the beginning of each copy cycle. If you have one subscription set for the Apply qualifier, start the Apply program using **opt4one**=y so that the Apply program caches in memory information about the subscription set members and columns and reuses it. When you optimize the Apply program for one subscription set, the Apply program uses less CPU, and you improve throughput rates.

Important: When you use **opt4one**=y and you add a member to a set or otherwise modify a set, you must stop the Apply program and start it again so that the Apply program picks up the changes in the control tables.

pwdfile (Linux, UNIX, Windows)

Default: **pwdfile**=asnpwd.aut

If your data is distributed across servers, you can store user IDs and passwords in an encrypted password file so that the Apply program can access data on remote servers. For more information, see "Storing user IDs and passwords for replication (Linux, UNIX, Windows)" on page 22.

sleep (Linux, UNIX, Windows, z/OS)

Default: **sleep**=y

The **sleep** parameter specifies whether the Apply program continues running in sleep mode or terminates after it processes eligible subscription sets.

By default, the Apply program starts with **sleep**=y. It checks for eligible subscription sets. If it finds an eligible subscription set, it processes it and continues looking for another eligible set. Apply continues to process eligible sets if it finds them. When it cannot find any more eligible sets, the Apply program continues running in sleep mode and it ″wakes up″ periodically to check if any subscription sets are eligible. Usually you want to start the Apply program in this way because you want updates applied over time and you expect the Apply program to be up and running.

Note: The **sleep** parameter is ignored if **copyonce** is specified.

When you start the Apply program with **sleep**=n, the Apply program checks for eligible subscription sets and processes them. It continues processing eligible subscription sets until it can't find any more eligible sets, and it repeats the process for eligible sets until there is no more data to replicate; then, the Apply program terminates. Typically you want to use **sleep**=n in a mobile environment or in a test environment where you want the Apply program to run *only* if it finds eligible subscription sets, and then you want it to terminate. You don't want the Apply program to wait in sleep mode and wake up periodically to check for more eligible sets. In these environments you want to control when Apply runs rather than have it run endlessly.

Tip: Use **copyonce**=y instead of **sleep**=n if you want to process each subscription set only once.

spillfile (Linux, UNIX, Windows, z/OS)

Default (Linux, UNIX, Windows): **spillfile**=disk

Default (z/OS): **spillfile**=MEM

Apply retrieves data from the source tables and places it in a spill file on the system where the Apply program is running.

On Linux, UNIX, and Windows operating systems, the only valid setting for **spillfile** is disk because spill files are *always* on disk in the location specified by the **apply_path** parameter.

On z/OS operating systems, including USS, the spill file is stored in memory by default. If you specify to store the spill file on disk, the Apply program uses the specifications on the ASNASPL DD statement to allocate spill files. If the ASNASPL DD statement is not specified, it uses VIO.

sqlerrcontinue (Linux, UNIX, Windows, z/OS)

Default: **sqlerrcontinue**=n

The **sqlerrcontinue** parameter specifies how the Apply program should react to certain SQL errors.

By default, when the Apply program encounters any SQL error, it stops processing that subscription set and generates an error message. Typically you would use the default in your production environment.

If you are in a test environment, you can expect certain SQL errors to occur when inserting data into target tables. Sometimes those errors are acceptable to you, but they would cause the current subscription cycle to stop. In those situations, you

can start the Apply program using **sqlerrcontinue**=y so that it ignores those errors and does not rollback replicated data from that cycle. If the Apply program receives an SQL error when inserting data into a target table, it checks the values in the *apply_qualifier*.sqs file. If it finds a match, it writes the details about the error to an error file, *apply_qualifier*.err, and it continues processing. If the Apply program encounters an SQL error that is not listed in the *apply_qualifier*.sqs file, it stops processing the set and goes on to the next set.

Before you start the Apply program using the **sqlerrcontinue**=y option, you must create the *apply_qualifier*.sqs file and store it in the directory from which you invoke the Apply program. List up to 20 five-byte values, one after the other, in the file. If you change the contents of this file when the Apply program is running, stop the Apply program and start it again so that it recognizes the new values.

Example: Assume that you want the Apply program to continue processing a subscription set if a target table gets the following error (sqlstate/code):

42704/-803
Duplicate index violation

You would create an SQL state file that contains the following SQL state:

```
42704
```

If the SQL state is returned when updating the target table, the Apply program applies the changes to the other target tables within the set and creates an error file indicating both the error and the rejected rows.

Tip: Check the STATUS column of the Apply trail (IBMSNAP_APPLYTRAIL) table. A value of 16 indicates that the Apply program processed the subscription set successfully, but some of the allowable errors, which you defined in the *apply_qualifier*.sqs file, occurred.

term (Linux, UNIX, Windows, z/OS)

Default: **term**=y

The **term** parameter determines how the status of DB2 affects the operation of the Apply program.

By default, the Apply program terminates if DB2 terminates.

Use **term**=n if you want the Apply program to wait for DB2 to start, if DB2 is not active. On the z/OS operating system, if DB2 quiesces and Apply is active, the Apply program will stay active and will not reconnect until DB2 is started again. On Linux, UNIX, and Windows operating systems, if DB2 quiesces and Apply is active, the Apply program will stay active and will not reconnect until DB2 is out of quiesce mode.

Note: The **term** parameter is ignored if **copyonce** is specified.

trlreuse (Linux, UNIX, Windows, z/OS)

Default: **trlreuse**=n

The **trlreuse** parameter specifies whether or not the Apply trail (IBMSNAP_APPLYTRAIL) table should be reused (appended to) or overwritten when the Apply program starts.

By default, when the Apply program starts, it appends entries to the Apply trail table. This table contains the history of operations for all Apply instances at the Apply control server. It is a repository of diagnostic and performance statistics. Keep the default if you want the history of updates. In the following situations you might want the Apply program to empty the Apply trail table when it starts instead of appending to it (**trlreuse**=y):

- The Apply trail table is getting too large, and you want to clean it out to save space.
- You don't need the history that is stored in the table.

Tip: Instead of using **trlreuse**=y, you can use SQL processing after the Apply program successfully completes a subscription set (where **status**=0) to delete rows from the Apply trail table.

Starting the Apply program (OS/400)

You can start an instance of the Apply program to begin applying data to your targets.

After you start the Apply program, it runs continuously unless one of the following conditions are true:

- You started the program using the COPYONCE(*YES) start-up parameter.
- You specified ALWINACT(*NO) and there is no data to be processed.
- You stop the Apply program using the Replication Center or a command.
- The Apply program cannot connect to the Apply control server.
- The Apply program cannot allocate memory for processing.

Prerequisites:

Before you start the Apply program, ensure that your system is set up correctly:

- Connections are configured to all replication servers.
- You have the proper authorization.
- The control tables are created.
- The replication programs are configured.

Also, make sure that the following conditions are met:

- At least one active subscription set exists for the Apply qualifier and that subscription set contains one or more of the following items:
  - Subscription-set member
  - SQL statement
  - Procedure
- All condensed target tables must have a target key, which is a set of unique columns, either a primary key or unique index, that the Apply program uses to track which changes it replicates during each Apply cycle. (Non-condensed CCD tables do not have primary keys or unique indexes.)

Procedure:

Use one of the following methods to start the Apply program:

Replication Center
Use the Start Apply window. See the Replication Center help for details.

STRDPRAPY system command
See "STRDPRAPY: Starting Apply (OS/400)" on page 386 for details.

When you start the Apply program, you can use these default settings for the operational parameters:

Table 9. Default settings for Apply operational parameters (OS/400)

Operational parameter	Description of (**value*)
USER (*CURRENT)	The user who signed on to the system.
JOBD (*LIBL/QZSNDPR)	Product library name / job description.
APYQUAL (*USER)	Current user name (from above).
CTLSVR (*LOCAL)	Local RDB server name.
TRACE (*NONE)	Do not generate a trace.
FULLREFPGM (*NONE)	Do not run the ASNLOAD exit routine.
SUBNFYPGM (*NONE)	Do not run the ASNDONE exit routine.
INACTMSG (*YES)	When the Apply program begins an inactive period, it generates message ASN1044, which describes how long the program will be inactive.
ALWINACT (*YES)	Sleep if there is nothing to process.
DELAY (6)	Wait 6 seconds after an Apply cycle before processing again.
RTYWAIT (300)	Wait 300 seconds before retrying a failed operation.
COPYONCE (*NO)	Do not terminate after completing one copy cycle, continue processing.
TRLREUSE (*NO)	Do not empty the Apply trail (IBMSNAP_APPLYTRAIL) table when the Apply program starts.
OPTSNGSET (*NO)	Do not optimize performance of the Apply program for processing a single subscription set.

For more information about these operational parameters, including syntax diagrams, see "STRDPRAPY: Starting Apply (OS/400)" on page 386.

Changing the operating parameters in the Apply parameters table (Linux, UNIX, Windows, z/OS)

The Apply parameters (IBMSNAP_APPPARMS) table contains the operational parameters for the Apply program. When you start the Apply program, it uses values from this table for its default operational behavior, unless you provide new values using the start-up parameters.

Only one row is allowed for each *apply_qualifier*. If you want to change one or more of the default values, you can update columns instead of inserting rows. If you delete the row, the Apply program will still start using the shipped defaults, unless those defaults are overridden by the start-up parameters.

The Apply program reads this table only during startup; therefore, you should stop and start the Apply program if you want the Apply program to run with the new settings. Changing the Apply parameters table while the Apply program is running

will not change the operation of the Apply program. See Chapter 24, “Table structures for SQL replication,” on page 421 for descriptions of the columns in this table.

Stopping the Apply program

You can stop an instance of the Apply program. When you stop the Apply program, it no longer copies data to the target tables, and it updates information in the control tables to ensure that the program starts cleanly the next time that you start it.

Prerequisites:

The instance of the Apply program must be started.

Procedure:

Use one of the following methods to stop an instance of the Apply program:

Replication Center
Use the Stop Apply window. See the Replication Center help for details.

asnacmd stop system command (Linux, UNIX, Windows, z/OS)
See “asnacmd: Operating Apply” on page 271 for details.

ENDDPRAPY system command (OS/400)
See “ENDDPRAPY: Stopping Apply (OS/400)” on page 361 for details.

Modifying the ASNDONE exit routine (Linux, UNIX, Windows, z/OS)

This section describes how to customize the ASNDONE exit routine on Linux, UNIX, Windows, and z/OS operating systems.

If you start the Apply program with the **notify**=y parameter, the Apply program calls the ASNDONE exit routine after it finishes processing subscriptions, regardless of whether the subscriptions were processed successfully. The following list describes some examples of how you might modify the ASNDONE exit routine to use it in your replication environment:

- Use the exit routine to examine the UOW table for rejected transactions and initiate further actions (for example, send e-mail automatically to the replication operator, issue a message, or generate an alert) if a rejected transaction is detected.
- Use the exit routine to deactivate a failed subscription set so that the Apply program avoids retrying that subscription set until it is fixed. To detect a failed subscription set, modify the exit routine to look for STATUS= -1 in the Apply trail (IBMSNAP_APPLYTRAIL) table. To deactivate the subscription set, configure the exit routine so that it sets ACTIVATE=0 in the subscription sets (IBMSNAP_SUBS_SET) table.
- Use the exit routine to manipulate data after it is applied for *each* subscription set. (Alternatively, you can define run-time processing statements using SQL statements or stored procedures that run before or after the Apply program processes a *specific* subscription set.)

Procedure:

To use a modified version of the ASNDONE sample exit routine:

1. Modify the ASNDONE routine to meet your requirements.

   Linux, UNIX, Windows: See the PROLOG section of the sample program (\sqllib\samples\repl\asndone.smp) for information about how to modify this exit routine.

   z/OS: See the PROLOG section of the sample program SASNSAMP(ASNDONE).
2. Compile, link, and bind the program and place the executable in the appropriate directory.
3. Start the Apply program with the **notify**=y parameter to call the ASNDONE exit routine.

Modifying the ASNDONE exit routine (OS/400)

This section describes how to customize the ASNDONE exit routine for an OS/400 environment.

If you start the Apply program with the SUBNFYPGM parameter set to the name of the ASNDONE exit routine, the Apply program calls the ASNDONE exit routine after it finishes processing subscriptions, regardless of whether the subscriptions were processed successfully. The following list describes some examples of how you might modify the ASNDONE exit routine to use it in your replication environment:

- Use the exit routine to examine the UOW table for rejected transactions and initiate further actions (for example, send e-mail automatically to the replication operator, issue a message, or generate an alert) if a rejected transaction is detected.
- Use the exit routine to deactivate a failed subscription set so that the Apply program avoids retrying that subscription set until it is fixed. To detect a failed subscription set, modify the exit routine to look for STATUS= -1 in the Apply trail (IBMSNAP_APPLYTRAIL) table. To deactivate the subscription set, configure the exit routine so that it sets ACTIVATE=0 in the subscription sets (IBMSNAP_SUBS_SET) table.
- Use the exit routine to manipulate data after it is applied for *each* subscription set. (Alternatively, you can define run-time processing statements using SQL statements or stored procedures that run before or after the Apply program processes a *specific* subscription set. See "Enhancing data using stored procedures or SQL statements" on page 98 for details.)

Procedure:

To use a modified version of the ASNDONE sample exit routine:

1. Modify the ASNDONE exit routine to meet your site's requirements.

   The following table indicates where you can find the source code for this routine in C, COBOL, and RPG languages:

Compiler language	Library name	Source file name	Member name
C	QDP4	QCSRC	ASNDONE
COBOL	QDP4	QCBLLESRC	ASNDONE
RPG	QDP4	QRPGLESRC	ASNDONE

   When modifying the program, consider these activation group concerns:

If the program is created to run with a new activation group: The Apply program and the ASNLOAD program will not share SQL resources, such as relational database connections and open cursors. The activation handling code in the OS/400 operating system frees any resources allocated by the ASNLOAD program before control is returned to the Apply program. Additional resource is used every time that the Apply program calls the ASNLOAD program.

If the program is created to run in the caller's activation group: It shares SQL resources with the Apply program. Design the program so that you minimize its impact on the Apply program. For example, the program might cause unexpected Apply program processing if it changes the current relational database connection.

If the program is created to run in a named activation group: It does not share resources with the Apply program. Use a named activation group to avoid the activation group overhead every time the ASNLOAD program is called. Run-time data structures and SQL resources can be shared between invocations. Application clean-up processing is not performed until the Apply program is ended, so design the subscription notify program to ensure that it does not cause lock contention with the Apply program by leaving source tables, target tables, or control tables locked when control is returned to the Apply program.

2. Compile, link, and bind the program, and place the executable in the appropriate directory.
3. Start the Apply program and specify the name of the ASNDONE program using the parameter SUBNFYPGM on the **STRDPRAPY** command. For example, if the program is named ASNDONE_1 and resides in library APPLIB, use the following command:

   ```
   SUBNFYPGM(APPLIB/ASNDONE_1)
   ```

Refreshing target tables using the ASNLOAD exit routine

By default, the Apply program does not use the ASNLOAD exit routine when it performs a full refresh for each target table in a subscription set. It does a full select against the source table, brings the data to a spill file on the server where the Apply program is running, and uses INSERT statements to populate the target table. If you have large source tables, you might want to use the ASNLOAD exit routine instead to copy the data more efficiently to the target during a full refresh.

The ASNLOAD exit routine is shipped as a sample exit routine in both a source format and a compiled format on Linux, UNIX, Windows, and z/OS operating systems; on OS/400 operating systems, ASNLOAD is shipped in a source format only. The sample exit routine differs on each DB2 platform to take advantage of the utility options offered on that platform.

If an error occurs when the Apply program calls the ASNLOAD exit routine, the Apply program issues a message, stops processing the current subscription set, and processes the next subscription set.

Prerequisites:

Ensure that the following prerequisites are met before you use the ASNLOAD exit routine:

- The target-table columns match both the order *and* data type of the source tables.
- The target table contains only columns that are part of the replication mapping.

Restrictions:

- On the z/OS operating system, the ASNLOAD exit routine calls the cross loader function that is available with the DB2 V7 (or higher) Utilities Suite.
- On the Linux, UNIX, and Windows operating system, the ASNLOAD exit routine works with the export, import, and load utilities, including the load from cursor function. Load from cursor is the default option used by the ASNLOAD exit if the source for a subscription-set member is actually a nickname, or if the target database is the same as the source database. The load from cursor function can also be used with DB2 data sources, if the following actions have been performed:
  - A nickname for the source table was created in the target database.
  - Columns in the IBMSNAP_SUBS_MEMBR table for the subscription-set member were set to indicate that the load from cursor function is to be used. The value of these columns can be set using the Replication Center.
    - The LOADX_TYPE has been set to indicate the load from cursor function should be used.
    - The source nickname information has been specified in the LOADX_SRC_N_OWNER and LOADX_SRC_N_TABLE columns in the IBMSNAP_SUBS_MEMBR table for the subscription-set member that includes the source table.

The following sections describe how to use the ASNLOAD exit routine on various operating systems:

- "Refreshing target tables with the ASNLOAD exit routine (Linux, UNIX, Windows)"
- "Refreshing target tables with the ASNLOAD exit routine (z/OS)" on page 137
- "Refreshing target tables with the ASNLOAD exit routine (OS/400)" on page 140

Refreshing target tables with the ASNLOAD exit routine (Linux, UNIX, Windows)

The ASNLOAD exit routine offers many utility options, such as using the DB2 EXPORT utility with either the DB2 IMPORT utility or the DB2 LOAD utility, or using the new LOAD FROM CURSOR utility. When you invoke the sample exit routine, by default it chooses which utility to use based on the source server, target server, and run-time environment.

You can use the compiled exit routine, you can configure its behavior by customizing the replication configuration, or you can customize the exit code itself. You can customize the replication configuration by either updating columns in the subscription members (IBMSNAP_SUBS_MEMBR) table or by updating a sample configuration file (asnload.ini).

Procedure:

To use the ASNLOAD routine as provided, start the Apply program using the **loadxit**=y parameter.

To use a modified version of the ASNLOAD sample exit routine:

1. Modify the ASNLOAD routine to meet your site's requirements. See the PROLOG section of the sample program (\sqllib\samples\repl\asnload.smp) for information about how to modify this exit routine.

Important: The sample source uses user ID and password combinations from the asnload.ini file. If the asnload.ini file does not have a user ID and password for a particular server, or if the asnload.ini file is not available, connect without the user/using phrases will be made.

2. Compile, link, and bind the program and place the executable in the appropriate directory.
3. Set LOADX_TYPE to 2 for members that will be populated using the code you provide. For more information, see "Customizing ASNLOAD exit behavior (Linux, UNIX, Windows, z/OS)" on page 138.
4. Start the Apply program with the **loadxit**=y parameter to call the ASNLOAD exit routine.

To configure input to the ASNLOAD exit routine, see "Using the configuration file for ASNLOAD (Linux, UNIX, Windows)" on page 139.

Files generated by the ASNLOAD exit routine:

These files are stored in the **apply_path** directory for the Apply instance that invoked the ASNLOAD exit routine.

- asnload *apply_qualifier*.trc

  This file contains trace information if the trace is turned on. The ASNLOAD exit routine creates this file. If the file exists, information is appended to the file.
- asnload *apply_qualifier*.msg

  This file contains general exit failure, warning, and informational messages, including load statistics. The ASNLOAD exit routine creates this file. If the file exists, information is appended to the file.
- asnaEXPT *apply_qualifier*.msg

  This file contains error, warning, or informational messages issued by the DB2 EXPORT utility. The ASNLOAD exit routine creates this file. If the file exists, information is appended to the file.
- asnaIMPT *apply_qualifier*.msg

  This file contains error, warning, or informational messages issued by the DB2 IMPORT utility. The ASNLOAD exit routine creates this file. If the file exists, information is appended to the file.
- asnaLOAD *apply_qualifier*.msg

  This file contains error, warning, or informational messages issued by the DB2 LOAD utility. The ASNLOAD exit routine creates this file. If the file exists, information is appended to the file.

Refreshing target tables with the ASNLOAD exit routine (z/OS)

The ASNLOAD exit routine calls the LOAD utility, which does cursor-based fetches to get data from the source and loads the data to the target. The ASNLOAD exit routine uses LOAD with LOG NO and resets the COPYPEND status of the table space. You can modify the sample ASNLOAD source code to change the load options. The source consists of two header files and three C++ programs.

Procedure:

To use the ASNLOAD routine as provided, start the Apply program using the **loadxit**=y parameter.

To use a modified version of the ASNLOAD sample routine:

1. Modify the ASNLOAD routine to meet your site's requirements. See the PROLOG section of the sample program SASNSAMP(ASNLOAD) for information about how to modify this exit routine.
2. Compile, link, and bind the program and place the executable in the appropriate directory. Add the ASNLOAD package to the Apply plan.
   a. Make sure that the following conditions are met:
      - DB2 Universal Database for z/OS and OS/390 Version 7 or later, with utility support, is installed.
      - DSNUTILS stored procedure is running. DSNUTILS must run in a WLM environment. For more information about using DSNUTILS, see the *DB2 Universal Database for OS/390 and z/OS Utility Guide and Reference*, SC26-9945 for more information.
   b. Use the sample zmak file (SASNSAMP(ASNCMPLD)) to compile and linkedit the ASNLOAD user exit program in USS.
   c. Bind the ASNLOAD exit routine with DSNUTILS and the Apply package.

      The sample ASNLOAD runs load with LOG NO and then repairs the table space to set nocopypend. It does not back up the table spaces. By default, ASNLOAD creates two temporary files under the user ID that is running the instance of the Apply program, unless the **apply_path** parameter with the APPLY_PATH=// option is specified for that Apply instance. If this is the case, then two temporary files will be created under the high level qualifier specified in APPLY_PATH. It also creates a file that contains all the information regarding the load.
3. Set loadx_type = 2 for members that will be populated using the code you provided.
4. Start the Apply program with the **loadxit**=y parameter to call the ASNLOAD exit routine.

To configure input to the ASNLOAD exit routine, see "Using the configuration file for ASNLOAD (Linux, UNIX, Windows)" on page 139.

Files generated by the ASNLOAD exit routine: These files are stored in the **apply_path** directory or HLQ for the Apply instance that invoked the ASNLOAD exit routine.

- *userid*.*apply_qual*.LOADMSG

  This file contains failure, warning, and informational messages, including load statistics. The ASNLOAD exit routine creates this file. If the file exists, information is appended to the file.
- *userid*.*apply_qual*.LOADTRC

  This file contains trace information if the trace is turned on. The ASNLOAD exit routine creates this file. If the file exists, information is appended to the file.

Customizing ASNLOAD exit behavior (Linux, UNIX, Windows, z/OS)

You can configure the behavior of the ASNLOAD exit routine by customizing the replication configuration, or you can customize the exit code itself. You can customize the replication configuration by either updating columns in the subscription member (IBMSNAP_SUBS_MEMBR) table or by updating a configuration file.

Using the subscription members table

You can use columns in the subscription member (IBMSNAP_SUBS_MEMBR) table to customize the behavior of the ASNLOAD exit routine. Use LOADX_TYPE to select the load option. The valid values for LOADX_TYPE are:

null (default)
On z/OS: Use the load from cursor function.

On Linux, UNIX, Windows: The ASNLOAD exit routine determines the most appropriate utility (option 3, 4, or 5).

1 Do not call ASNLOAD exit routine for this member.

Set LOADX_TYPE to 1 if you do not want the ASNLOAD exit routine to be called for that member.

2 Provide your own exit logic.

If you want to provide your own logic in the ASNLOAD exit routine, set LOADX_TYPE to 2 for those subscription set members that you want populated by the ASNLOAD exit routine. If you set LOADX_TYPE to 2 but you do not provide exit logic, the exit will fail.

3 Use the load from cursor function.

On Linux, UNIX, and Windows operating systems, the load from cursor function requires a SELECT statement to fetch the data that is to be loaded to the target table (the target table must reside in a local database). This statement can refer either to a DB2 table or to a nickname, and the setup must be as follows:

If you are replicating from a non-IBM source to a DB2 table where the registered source nickname is on a different database from the target database or if you are replicating from a DB2 table to another DB2 table and the source database is different from the target database, you need to do the following steps:

1. Create a nickname for the source table(s) in the target server database.
2. Update the nickname owner and the table name columns (LOADX_SRC_N_OWNER and LOADX_SRC_N_TABLE) of the subscription member (IBMSNAP_SUBS_MEMBR) table.

If you are replicating from a DB2 table to another DB2 table and the source and the target database are the same, or if you are replicating from a non-IBM source to a DB2 table where the registered source nickname is on the same database as the target database, no additional actions are needed to use the load from cursor function.

4 (Linux, UNIX, and Windows only)
Use a combination of the EXPORT utility and the LOAD utility.

5 (Linux, UNIX, and Windows only)
Use a combination of the EXPORT utility and the IMPORT utility.

Using the configuration file for ASNLOAD (Linux, UNIX, Windows)

You can use an optional configuration file to configure input to the ASNLOAD exit routine. This file is not required for ASNLOAD to run.

On Linux, UNIX, and Windows operating systems, the configuration file must have the file name asnload.ini. The ASNLOAD exit routine looks for this optional configuration file in the **apply_path** directory. Edit the sample file

sqllib/samples/repl/asnload.ini and store it in the **apply_path** directory for the Apply instance that invoked the ASNLOAD exit routine.

Refreshing target tables with the ASNLOAD exit routine (OS/400)

Use the exit routine instead of the Apply program to perform a full refresh more efficiently. For example, if you are copying every row and every column from a source table to a target table, you can design a full-refresh exit routine that uses a Distributed Data Management (DDM) file and the Copy File (**CPYF**) CL command to copy the entire file from the source table to the target table.

Procedure:

To refresh target tables using the ASNLOAD exit routine, start the Apply program using the FULLREFPGM parameter.

To use a modified version of the ASNLOAD sample routine:

1. Modify the ASNLOAD exit routine to meet your site's requirements. See the PROLOG section of the sample program for information about how to modify this exit routine. The source is available in C, COBOL, and RPG languages:

Compiler language	Library name	Source file name	Member name
C	QDP4	QCSRC	ASNLOAD
COBOL	QDP4	QCBLLESRC	ASNLOAD
RPG	QDP4	QRPGLESRC	ASNLOAD

2. Compile, link, and bind the program and place the executable in the appropriate directory.

   To avoid interference with the Apply program, compile the exit routine so that it uses a new activation group (not the activation group of the caller).

   You can compile the exit routine with a named activation group or with a new activation group. To get better performance, use a named activation group. With a named activation group, the exit routine must commit or roll back changes as needed. The Apply program will not cause changes to be committed or rolled back (unless it ends). The exit routine should either explicitly commit changes, or it should be compiled to implicitly commit changes when it completes. Any uncommitted changes when the exit routine completes are not committed until either:

   - The Apply program calls another exit routine with the same activation group.
   - The job started for the Apply program ends.
3. Start the Apply program with the FULLREFPGM parameter set to the name of the ASNLOAD program.

   When you start the Apply program, it uses the ASNLOAD exit routine that you specified. If you want it to use another ASNLOAD exit routine, end the Apply program and start it again.

When you run the ASNLOAD exit routine, it refreshes all the target tables, table by table.

Related tasks:

- Chapter 20, "Operating the SQL replication programs (z/OS)," on page 405

- Chapter 21, "Using the Windows Service Control Manager to issue system commands for SQL replication (Windows)," on page 409

Related reference:

- "asnscrt: Creating a DB2 replication service to start the replication programs" on page 302
- "asnacmd: Operating Apply" on page 271
- "asnapply: Starting Apply" on page 276
- "ENDDPRAPY: Stopping Apply (OS/400)" on page 361
- "STRDPRAPY: Starting Apply (OS/400)" on page 386

Chapter 11. Monitoring replication with the Replication Alert Monitor

Monitoring replication with the Replication Alert Monitor—Overview

You can use the Replication Alert Monitor to monitor an SQL replication, Q replication, or event publishing environment. The following topics explain how the Replication Alert Monitor works:

- "The Replication Alert Monitor"
- "Alert conditions and notifications for the Replication Alert Monitor—Overview" on page 145
- "Setting up the Replication Alert Monitor" on page 151
- "Operating the Replication Alert Monitor" on page 153

The Replication Alert Monitor

The Replication Alert Monitor is a program that checks the status of your replication environment. When the Replication Alert Monitor is running, it the automatically checks the status of replication and notifies you about certain conditions that occur in you replication environment. For example, in SQL replication, the Replication Alert Monitor can notify you when any Apply program terminates. Similarly, in Q replication, the Replication Alert Monitor can notify you when any Q Capture program deactivates a Q subscription.

You can check the status of your replication environment manually by using the following methods:

- You can view Replication Center windows that report statistics about the Capture, Q Capture, Apply, and Q Apply programs.
- You can run select statements on your control tables to view statistics about the operation of these programs.

In both of these cases, the statistics are available at any time, but you must retrieve them manually. Manually checking statistics about your replication environment is not an effective way to monitor replication activity continuously on many servers. The Replication Alert Monitor will *automatically* monitor your replication environment across all of your operating systems. It checks replication on your servers and *automatically* alerts you to conditions that require attention.

The Replication Alert Monitor can monitor the following replication environments:

- SQL replication
- Q replication
- Event publishing

You can use the Replication Alert Monitor to monitor the following replication programs:

- Capture program (SQL replication)
- Apply program (SQL replication)
- Q Capture program (Q replication or event publishing)
- Q Apply program (Q replication)

You can configure the Replication Alert Monitor in one of two ways: you can run one monitor, or you can run multiple monitors. Typically you use one monitor when you have few replication programs to monitor. Use additional monitors to monitor many replication programs, prioritize the monitoring of certain programs, or split up the monitoring workload. Setting up multiple monitors is like cloning the Replication Alert Monitor. You create distinct, independent, but similar Replication Alert Monitors, called monitors, to monitor the servers in your system. These monitors do not communicate with each other, but they each send alerts about the servers that they are monitoring. So collectively, these monitors send you alerts for all of the servers in your system. When you set up multiple monitors, the control information for each monitor is stored on the server that it is assigned to monitor.

If you set up a single monitor, all the control information is stored on one server. Each monitor can monitor multiple replication programs, but the monitor checks for alerts on each server one at a time. It must check all of the other servers that it monitors before it returns to any one server.

Whether you set up many monitors, or whether you configure one monitor, any server that contains control information is called a Monitor control server.

The following terms describe components of the Replication Alert Monitor:

Monitor
A monitor is one instance or occurrence of the Replication Alert Monitor. You can set up a monitor to check the status of the replication programs that are running on a server or servers. Each monitor checks the replication activity on the server, or servers, that it is assigned to.

Monitor qualifier
A monitor qualifier is a name that you specify for a monitor. Every monitor has a unique monitor qualifier.

Monitor control server
A monitor control server is any server containing control information for the Replication Alert Monitor.

Alerts Alerts are notices that tell you about events and conditions in your replication environment. The Replication Alert Monitor sends alerts via e-mail or pager.

Alert conditions
Alert conditions are conditions of the replication environment that cause the Replication Alert Monitor to send alerts. There are three kinds of alert conditions: alert conditions that are triggered by status, alert conditions that are triggered by events, and alert conditions that are triggered by thresholds.

Alert conditions that are triggered by status
Status alert conditions inform you about the state of the replication programs. For example, if you specify the APPLY_STATUS alert condition, the Replication Alert Monitor will send an alert if an Apply program is not running.

Alert conditions that are triggered by events
Event alert conditions tell you when specific events in replication happen. For example, if you specify the QAPPLY_ERRORS alert

condition, the Replication Alert Monitor will send an alert anytime the Q Apply program records an error in the IBMQREP_APPLYTRACE table.

Alert conditions that are triggered by thresholds
Threshold alert conditions tell you when a threshold has been exceed in your replication environment. For example, if you specify the QCAPTURE_MEMORY alert condition, the Replication Alert Monitor will notify you anytime the Q Capture program uses more memory than its threshold allows.

Contacts
A contact is an e-mail address or a pager address for a person to receive alerts from the Replication Alert Monitor.

Contact groups
A contact group is a collection of contacts that receive the same alerts.

The Replication Alert Monitor monitors servers on DB2® UDB for Linux, UNIX®, Windows®, or z/OS™ operating systems. You can use the Replication Alert Monitor to monitor DB2 UDB replication programs whose control tables are at DB2 replication Version 8 architecture or later.

Restrictions:

- In the case of iSeries™ servers, the Replication Alert Monitor must run on a Linux, UNIX, or Windows server. In this case the Replication Alert Monitor must monitor the iSeries server remotely.
- You cannot set up Monitor control servers on DB2 for iSeries servers.
- The Replication Alert Monitor does not monitor triggers that are associated with non-DB2 relational databases that are used as sources in a federated database system.

Related concepts:

- "Monitoring replication with the Replication Alert Monitor—Overview" on page 143

Alert conditions and notifications for the Replication Alert Monitor

Alert conditions and notifications for the Replication Alert Monitor—Overview

The following topics contain information about alert conditions for the Replication Alert Monitor:

- "Alert conditions for the Replication Alert Monitor" on page 146
- "E-mail notifications for replication alert conditions" on page 149
- "The ASNMAIL exit routine for sending alerts in replication" on page 150

Related concepts:

- "Monitoring replication with the Replication Alert Monitor—Overview" on page 143

Alert conditions for the Replication Alert Monitor

Alert conditions are conditions of the replication environment that cause a monitor to send alerts. Alerts are messages that describe the status, event or threshold that triggers an alert condition. Some alerts also report relevant parameter values. For example, the message for the QCAPTURE_MEMORY alert condition reports the amount of memory that the Q Capture program is using and the memory threshold value that was exceeded.

The following tables describe alert conditions that you can use to monitor your replication environment.

- "Alert conditions for the Q Capture program"
- "Alert conditions for the Q Apply program"
- "Alert conditions for the Capture program" on page 147
- "Alert conditions for the Apply program" on page 148

Alert conditions for the Q Capture program

Table 10 describes the alert conditions for the Q Capture program.

Table 10. Alert conditions for the Q Capture program

Alert condition	Description
QCAPTURE_STATUS	The Replication Alert Monitor sends an alert when a Q Capture program is not running.
QCAPTURE_ERRORS	The Replication Alert Monitor sends an alert when it finds a row with the value of 'ERROR' in the OPERATION column of the IBMQREP_CAPTRACE table.
QCAPTURE_WARNINGS	The Replication Alert Monitor sends an alert when it finds a row with the value of 'WARNING' in the OPERATION column in the IBMQREP_CAPTRACE table.
QCAPTURE_LATENCY	Q Capture latency measures the difference between the time that data was written to the database and the time that the Q Capture program passes it on. The Replication Alert Monitor sends an alert when the Q Capture latency exceeds the threshold that you specify. Q Capture latency is measured in seconds.
QCAPTURE_MEMORY	The Replication Alert Monitor sends an alert when a Q Capture program uses more memory than the threshold that you specify. Memory is measured in megabytes.
QCAPTURE_TRANSIZE	The Replication Alert Monitor sends an alert when a transaction that the Q Capture program is processing uses more memory than the threshold that you specify. Memory is measured in megabytes.
QCAPTURE_SUBSINACT	The Replication Alert Monitor sends an alert when a Q Capture program deactivates a Q subscription.

Alert conditions for the Q Apply program

Table 11 describes the alert conditions for the Q Apply program.

Table 11. Alert conditions for the Q Apply program

Alert condition	Description
QAPPLY_STATUS	The Replication Alert Monitor sends an alert when a Q Apply program is not running.

Table 11. Alert conditions for the Q Apply program (continued)

Alert condition	Description
QAPPLY_ERRORS	The Replication Alert Monitor sends an alert when it finds a row with the value of 'ERROR' in the OPERATION column in the IBMQREP_APPLYTRACE table.
QAPPLY_WARNINGS	The Replication Alert Monitor sends an alert when it finds a row with the value of 'WARNING' in the OPERATION column in the IBMQREP_APPLYTRACE table.
QAPPLY_LATENCY	Q Apply latency measures the time it takes for a transaction to be applied to a target table after the Q Apply program gets the transaction from a receive queue. The Replication Alert Monitor sends an alert when the Q Apply latency exceeds the threshold that you specify. Q Apply latency is measured in milliseconds.
QAPPLY_EELATENCY	Q Apply end-to-end latency measures the total time that replication requires to capture changes and apply those changes to a target database. The Replication Alert Monitor sends an alert when the Q Apply end-to-end latency exceeds the threshold that you specify. Q Apply end-to-end latency is measured in seconds.
QAPPLY_MEMORY	The Replication Alert Monitor sends an alert when a Q Apply program uses more memory than the threshold that you specify. Memory is measured in megabytes.
QAPPLY_EXCEPTIONS	The Replication Alert Monitor sends an alert when it finds an exception in the operation of a Q Apply program.
QAPPLY_SPILLQDEPTH	The Replication Alert Monitor sends an alert when the fullness of the spill queue exceeds the threshold that you specify. Fullness is expressed as a percentage.
QAPPLY_QDEPTH	The Replication Alert Monitor sends an alert when the fullness of any queue exceeds the threshold that you specify. Fullness is expressed as a percentage.

Alert conditions for the Capture program

Table 12 describes the alert conditions for the Capture program.

Table 12. Alert conditions for the Capture program

Alert condition	Description
CAPTURE_STATUS	The Replication Alert Monitor sends an alert when a Capture program is not running.
CAPTURE_ERRORS	The Replication Alert Monitor sends an alert when it finds a row with the value of 'ERROR' in the OPERATION column of the IBMSNAP_CAPTRACE table.
CAPTURE_WARNINGS	The Replication Alert Monitor sends an alert when it finds a row with the value of 'WARNING' in the OPERATION column of the IBMSNAP_CAPTRACE table.
CAPTURE_LASTCOMMIT	The Replication Alert Monitor sends an alert when the time that elapsed from the last commit of a Capture program exceeds the threshold that you specify. Time elapsed is measured in seconds.
CAPTURE_CLATENCY	The current capture latency measures the difference between the time that data was written to the database and the time that the Q Capture program passes it on. The Replication Alert Monitor sends an alert when the current Capture latency exceeds the threshold that you specify.

Table 12. Alert conditions for the Capture program (continued)

Alert condition	Description
CAPTURE_HLATENCY	Historic Capture latency is a composite of every Capture latency measurement since the monitor last checked a server for alert conditions. The Replication Alert Monitor sends an alert when the historic Capture latency exceeds the threshold that you specify.
CAPTURE_MEMORY	The Replication Alert Monitor sends an alert when a Capture program uses more memory than the threshold that you specify. Memory is measured in megabytes.

Alert conditions for the Apply program

Table 13 describes the alert conditions for the Apply program.

Table 13. Alert Conditions for the Apply program

Alert condition	Description
APPLY_STATUS	The Replication Alert Monitor sends an alert when an Apply program is not running.
APPLY_SUBSFAILING	The Replication Alert Monitor sends an alert when a subscription fails.
APPLY_SUBSINACT	The Replication Alert Monitor sends an alert when a subscription is deactivated.
APPLY_ERRORS	The Replication Alert Monitor sends an alert when it finds a row with the value of 'ERROR' in the OPERATION column in the IBMSNAP_APPLYTRACE table
APPLY_WARNINGS	The Replication Alert Monitor sends an alert when it finds a row with the value of 'WARNING' in the OPERATION column in the IBMSNAP_APPLYTRACE table
APPLY_FULLREFRESH	The Replication Alert Monitor sends an alert when there is a full refresh.
APPLY_REJTRANS	The Replication Alert Monitor sends an alert when a transaction is rejected in a subscription set.
APPLY_SUBSDELAY	The Replication Alert Monitor sends an alert when the delay in processing a subscription is longer than the threshold that you specify.
APPLY_REWORKED	The Replication Alert Monitor sends an alert when the Apply program reworks more rows in a subscription set than the threshold that you specify.
APPLY_LATENCY	Apply end-to-end latency measures the total time that replication requires to capture changes and apply those changes to a target database. The Replication Alert Monitor sends an alert when the Apply end-to-end latency exceeds the threshold that you specify. Apply end-to-end latency is measured in seconds.

Related concepts:

- "Monitoring replication with the Replication Alert Monitor—Overview" on page 143
- "The Replication Alert Monitor" on page 143

Related tasks:

- "Selecting alert conditions for the Replication Alert Monitor" on page 155

E-mail notifications for replication alert conditions

The Replication Alert Monitor can send an e-mail when an alert condition occurs. The content of the e-mail notification depends on whether the e-mail address that you provided is for a pager. The following examples show the type of information that you can expect in each case, for one set of alerts. The e-mail that is sent to non-pager devices shows the time when each alert condition occurred at the specific server. It also shows the number of times that each alert condition occurred and the associated message. The e-mail that the Replication Alert Monitor sends to pagers contains a summary of the parameters that triggered the alert instead of a complete message. If an alert condition occurred many times, the timestamp reflects the last time that the alert condition occurred.

Example e-mail notification to non-pager devices (SQL replication):

```
To:      repladmin@company.com
From:    replmon@server.com
Subject: Monitor: "MONQUAL" Alerts issued

ASN5129I MONITOR "MONQUAL". The Replication Alert Monitor on
server "WSDB" reports an e-mail alert

2002-01-20-10.00.00    1 ASN0552E Capture : "ASN" The program
encountered an SQL error. The server name is "CORP". The SQL
request is "PREPARE". The table name "PROD1.INVOICESCD".
The SQLCODE is "-204". The SQLSTATE is "42704". The SQLERRMC
is "PROD1.INVOICESCD". The SQLERRP is "readCD"

2002-01-20-10.05.00    2 ASN5152W Monitor "MONQUAL". The current
Capture latency exceeds the threshold value. The Capture control
server is "CORP". The schema is "ASN". The Capture
latency is "90" seconds. The threshold is "60" seconds

2002-01-20-10.05.00    4 ASN5154W Monitor "MONQUAL". The memory
used by the Capture program exceeds the threshold value. The
Capture control server is "CORP".  The schema is "ASN".
The amount of memory used is "34" bytes. The threshold is
"30" megabytes.
```

Example e-mail notification to pagers (SQL replication):

```
To:      repladmin@company.com
From:    replmon@server.com
Subject: Monitor: "MONQUAL" Alerts issued

MONQUAL - MONDB

2002-01-20-10.00.00 ASN0552E  1 CAPTURE-ERRORS - CORP - ASN
2002-01-20-10.05.00 ASN5152W  2 CAPTURE_CLATENCY - CORP - ASN - 90 - 60
2002-01-20-10.05.00 ASN5154W  4 CAPTURE_MEMORY - CORP - ASN - 34 - 30
```

In SQL replication, the Replication Alert Monitor groups alerts by Capture control servers and Apply control servers when it sends notifications. If a server is both a Capture control server and an Apply control server, then the Replication Alert Monitor groups all alerts for that server together.

In Q replication, the Replication Alert Monitor groups alerts by Q Capture servers and Q Apply servers when it sends notifications. If a server is both a Q Capture server and a Q Apply server, then the Replication Alert Monitor groups all alerts for that server together.

If the size of the e-mail notification exceeds the limit for the type of e-mail, the Replication Alert Monitor sends notification in multiple e-mails. The maximum size of a regular e-mail notification is 1024 characters. For a pager e-mail address the limit is 250 characters.

The ASNMAIL exit routine sends e-mail notifications for the Replication Alert Monitor. You can modify this exit routine to handle alerts differently. For example, you could have the ASNMAIL user exit routine store the alerts in a problem management system. See "The ASNMAIL exit routine for sending alerts in replication" for details.

Related concepts:

- "Alert conditions for the Replication Alert Monitor" on page 146

Related tasks:

- "Operating the Replication Alert Monitor" on page 153
- "Specifying notification criteria for selected alert conditions" on page 162

Related reference:

- "The ASNMAIL exit routine for sending alerts in replication" on page 150

The ASNMAIL exit routine for sending alerts in replication

The ASNMAIL exit routine handles notification. This exit routine takes the following input:

```
asnmail email_server to_address subject alert_message alert_message
```

Table 14 describes the inputs for the ASNMAIL exit routine.

Table 14. Inputs for the ASNMAIL exit routine

Input	Description
email_server	This is the address of an e-mail server that uses the SMTP protocol. This server address is passed from the **email_server** parameter specified in at the start of the **asnmon** command.
to_address	This is the e-mail address of the contact to be notified.
subject	This is the subject in the notification.
alert_message	This is the string that contains the alert message.

Instead of sending alerts by e-mail, you can modify the ASNMAIL exit routine to put the alerts elsewhere such as in a problem management system. The \sqllib\samples\repl\ directory contains a sample of the ASNMAIL exit routine. The asnmail.c sample contains the input parameters and directions for using the sample program.

Related concepts:

- "E-mail notifications for replication alert conditions" on page 149
- "Alert conditions for the Replication Alert Monitor" on page 146

Related tasks:

- "Operating the Replication Alert Monitor" on page 153

Setting up the Replication Alert Monitor

Setting up the Replication Alert Monitor

Your replication environment consists of the replication programs that run on servers and the control tables that support them. The Replication Alert Monitor monitors this environment for you.

Procedure:

Use this procedure once when you install the Replication Alert Monitor. To set up the Replication Alert Monitor to monitor your replication environment:

1. Evaluate your monitoring load. The more servers that you have to monitor, the greater the monitoring load.
2. Choose how you want to configure your monitors. You have two configuration options. You can run one monitor or multiple monitors. Use multiple monitors to:
   - **Monitor some replication programs more frequently than others.** Set up a monitor with a smaller monitor_interval to check replication programs for alert conditions more frequently. For example, you can assign one monitor to monitor one Capture server for the CAPTURE_WARNINGS alert condition every 15 minutes. You can also assign another monitor to monitor another Capture server for the CAPTURE_WARNINGS alert condition every 50 minutes.
   - **Monitor different applications separately.** Set up monitors for each replication application. For example, separate monitors can send alerts to different groups or help an administrator separate the alerts for two different applications. Similarly, separate monitors can be assigned to check for different alert conditions.
   - **Prioritize alert conditions.** For example, you might want to monitor the status of a Q Apply program every 10 minutes by using the QAPPLY_STATUS alert condition. But, you might also want to monitor the memory of the same Q Apply program every 300 minutes by using the QAPPLY_MEMORY alert condition.
3. Choose which servers to set up as monitor control servers. Choose one monitor control server for each monitor that you would like to set up.
4. Create control tables for each Monitor control server. See "Creating control tables for the Replication Alert Monitor" for details.
5. Define contact information for the Replication Alert Monitor. See "Defining contact information for the Replication Alert Monitor" on page 152 for details.

Related concepts:

- "Monitoring replication with the Replication Alert Monitor—Overview" on page 143

Creating control tables for the Replication Alert Monitor

Before you can use the Replication Alert Monitor, you must create monitor control tables. These tables store alert conditions, contact information, run-time parameters, and other metadata for the monitor. The server where you create the monitor control tables is called a Monitor control server.

The Monitor control server can be a DB2 UDB for Linux, UNIX, Windows, or z/OS server. In most cases you will need only one Monitor control server, but you can use multiple servers depending on your replication environment. For example, if you want monitors to run on the same system as the replication programs that they monitor, create one set of control tables for each local monitor on the server where the monitor runs.

Procedure:

To create control tables for the Replication Alert Monitor, use the Create Monitor Control Tables window in the Replication Center. To open the window, right-click the **Monitor Control Servers** folder and select **Create Monitor Control Tables**. See the online help for details.

Related concepts:

- "Monitoring replication with the Replication Alert Monitor—Overview" on page 143

Related reference:

- "List of tables at the Monitor control server" in the *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference*

Defining contact information for the Replication Alert Monitor

You must define contact information for the individuals or groups that you want to notify of alert conditions before you use the Replication Alert Monitor for the first time. You can change contact information after monitors are running.

Contact information is stored on Monitor control servers. Monitors running on the same Monitor control server can share contacts. If you have multiple Monitor control servers, you must define contacts for each server.

After you define contacts by specifying the e-mail address for and the name of each contact, you can put contacts into groups. For example, you could set up a contact group called `DB2 administrators` that contains the contact information for all your DB2 administrators. You can also copy contact and group information from one server to another.

Procedure:

To define contact information for the Replication Alert Monitor:

1. Create contacts and contact groups for the monitors on a Monitor control server.
   a. In the Replication Center, use the Create Contact window to define contact information.
   b. Optional: Create contact groups by using the Create Contact Group window in the Replication Center.

   To open the windows, expand the Monitor control server for which you want to add a contact or contact group, right-click the **Contacts** folder and select **Create Contact → Person** or **Create Contact → Group**. See the online help for details.
2. Optional: Copy contact information from one Monitor control server to another Monitor control server by using the Copy Contacts and Contact Groups window in the Replication Center.

To open the window, expand the Monitor control server on which the contacts or contact groups are located. Select the **Contacts** folder. In the contents pane, right-click the contacts or contact groups that you want to copy, and select **Copy**. See the online help for details.

Contacts that you create for the Replication Alert Monitor in the Replication Center cannot be used in other DB2 UDB centers such as the Task Center or the Health Center. Contacts created in other DB2 UDB centers cannot be used by the Replication Alert Monitor.

Related concepts:

- "E-mail notifications for replication alert conditions" on page 149

Related reference:

- "The ASNMAIL exit routine for sending alerts in replication" on page 150

Operating the Replication Alert Monitor

Operating the Replication Alert Monitor

You can monitor your replication environment by running the Replication Alert Monitor.

Prerequisites:

Before you can operate the Replication Alert Monitor, you must set up the Replication Alert Monitor. See "Setting up the Replication Alert Monitor" on page 151 for details.

Procedure:

You must perform the following tasks for each monitor. The following topics explain how to operate the Replication Alert Monitor:

1. Create the monitor. See "Creating monitors for replication or publishing" on page 154 for details.
2. Select alert conditions for the monitor. See "Selecting alert conditions for the Replication Alert Monitor" on page 155 for details.
3. Optional: Set parameters for the monitor. See "Setting parameters for the Replication Alert Monitor—Overview" on page 160 for details.
   - "Descriptions of the parameters that are used to operate the Replication Alert Monitor" on page 158
   - "Default values of the parameters that are used to operate the Replication Alert Monitor" on page 157
4. Start the Replication Alert Monitor. See "Starting monitors" on page 156 for details.

You can set parameters before you start a monitor, when you start a monitor, or while a monitor is running. To refresh the parameter settings for a running monitor, you must reinitialize the monitor. See "Reinitializing monitors" on page 156 for details.

You can also stop a running monitor. See "Stopping a monitor" on page 163 for details.

Related concepts:

- "Monitoring replication with the Replication Alert Monitor—Overview" on page 143

Creating monitors for replication or publishing

After you create monitor control tables, you can use the Create Monitor wizard in the Replication Center to create monitors and select the alert conditions that will be used to monitor your replication or publishing environment.

Prerequisites:

Before you create monitors, you must set up the Replication Alert Monitor. See "Setting up the Replication Alert Monitor" on page 151 for details.

Procedure:

To create a monitor:

1. In the Replication Center, open the Create Monitor wizard and specify the name of the monitor and the replication or publishing programs that the monitor will check for alert conditions:

   To open the wizard, expand the Monitor control server on which you want to create a monitor, right-click the **Monitors** folder, and select **Create**.

   a. On the Start page, specify a monitor qualifier. Then, specify the programs that you would like this monitor to check for alert conditions. You can also monitor subscription sets that are used in SQL replication.

   The wizard directs you to one or more of the following pages where you can select alert conditions, depending on which replication programs you want this monitor to check for alert conditions:

   - Select alert conditions for Q Capture programs
   - Select alert conditions for Q Apply programs
   - Select alert conditions for Capture programs
   - Select alert conditions for Apply programs
   - Select alert conditions for subscription sets

   For example, if you specified that you want to monitor Q Capture programs and Q Apply programs, then the Create Monitor wizard directs you to the Select alert conditions for Q Capture programs page and the Select alert conditions for Q Apply programs page.
2. From one of the pages that are listed above, open secondary dialogs where you can:
   a. Specify the programs or subscription sets that you want to monitor.
   b. Specify the alert conditions that you want to check for, and the parameters for the appropriate alert conditions. For example, you can set the **monitor_interval** parameter value to 60 to make the monitor check for alert conditions once every minute.
3. On the Summary page, click **Finish**.

See the online help for details.

Related concepts:

- "The Replication Alert Monitor" on page 143

Selecting alert conditions for the Replication Alert Monitor

The Replication Alert Monitor monitors the activity of the replication and publishing programs at the following times:

- Each monitor checks for alert conditions immediately when you start it.
- Each monitor checks for alert conditions periodically, at timed intervals that you specify.

When you create a monitor, you select the alert conditions that will prompt that monitor to send alerts. You can select alert conditions for each Q Capture program, Q Apply program, Capture program, Apply program, or subscription set that a monitor is monitoring.

You can also use the Replication Center to change alert conditions while the monitor is running. You do this by opening an existing monitor, changing the alert conditions, and then reinitializing the monitor.

Procedure:

To select alert conditions for the Replication Alert Monitor:

1. Use one or more of the following pages in the Create Monitor wizard in the Replication Center, depending on which program you chose to monitor:
   - Select alert conditions for Q Capture programs
   - Select alert conditions for Q Apply programs
   - Select alert conditions for Capture programs
   - Select alert conditions for Apply programs
   - Select alert conditions for subscription sets
2. Specify thresholds that are compatible with your environment.

   For example, if a Capture program is running with a commit interval of 30 seconds, specify a threshold for Capture latency that is greater than 30 seconds. Or, if you schedule an Apply program to process subscription sets every 10 minutes, set the threshold of the APPLY_SUBSDELAY alert condition to a value that is greater than 10 minutes.

To change alert conditions for the Replication Alert Monitor:

1. In the Replication Center, open the Alert Conditions window for a Q Capture program, Q Apply program, Capture program, Apply program, or subscription set. To open the windows:
   a. Expand the **Monitors** folder within the correct Monitor control server.
   b. Select a monitor.
   c. In the contents pane, right-click the Q Capture schema, Q Apply schema, Capture schema, Apply schema, or subscription set that you want to change alert conditions for.
   d. Select **Change**.
2. Change the alert conditions.
3. Reinitialize the monitor. See “Reinitializing monitors” on page 156 for details.

See the online help for details.

Related concepts:

- “Alert conditions for the Replication Alert Monitor” on page 146

- "The Replication Alert Monitor" on page 143

Starting monitors

You use the Replication Center to start a monitor. You can decide whether to run the monitor continuously or for only one monitor cycle. You can also set values for parameters, and enter the e-mail address of the person to contact if the monitor itself encounters an error while running.

Prerequisites:

- Create a monitor. See "Creating monitors for replication or publishing" on page 154 for details.
- Create a password file. See "asnpwd: Creating and maintaining password files" on page 299 for details.
- Make sure that you have the correct authorization to access the monitor control tables and servers where the programs that you want to monitor are running.

Procedure:

To start a monitor, use one of the following methods:

Replication Center
Use the Start Monitor window. To open the window, right-click the Monitor qualifier that identifies the monitor that you want to start, and select **Start Monitor**. See the online help for details.

asnmon system command
You can use the **asnmon** command to start a monitor and optionally specify startup parameters. See "asnmon: Starting a Replication Alert Monitor" on page 295 for details.

Windows Service Control Manager
You can set up the Windows Service Control Manager to run a monitor.

z/OS console or TSO
You can set up the Automatic Restart Manager (ARM) recovery system to start a monitor.

Related concepts:

- "The Replication Alert Monitor" on page 143
- "The Automatic Restart Manager (ARM) recovery system" in the *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference*

Related tasks:

- "Reinitializing monitors" on page 156
- "Stopping a monitor" on page 163
- "Scheduling the replication programs (Windows)" in the *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference*

Reinitializing monitors

You can reinitialize a monitor while it is running. Reinitializing a monitor causes it to recognize any updates that you have made to contacts, alert conditions, and parameter values. For example, reinitialize a monitor if you added a new e-mail address for a contact while the monitor is running.

Procedure:

To reinitialize a monitor, use one of the following methods:

Replication Center
Use the Reinitialize Monitor window to reinitialize a monitor. To open the window, right-click the Monitor qualifier that identifies the monitor that you want to reinitialize, and select **Reinitialize Monitor**. See the online help for details.

asnmcmd system command
You can use the **asnmcmd reinit** command to reinitialize a running monitor. See "asnmcmd: Working with a running Replication Alert Monitor" on page 292 for details.

Related tasks:

- "Starting monitors" on page 156
- "Stopping a monitor" on page 163

Default values of the parameters that are used to operate the Replication Alert Monitor

You can change the behavior of the Replication Alert Monitor by setting values for the Replication Alert Monitor parameters. Table 15 shows the default value for each parameter.

Table 15. Default values for Replication Alert Monitor operating parameters

Operational parameter	Default value
alert_prune_limit	10080 minutes
autoprune	Y
email_server	no default value
max_notification_minutes	60 minutes
max_notifications_per_alert	3
monitor_errors	no default value
monitor_interval	300 seconds
monitor_limit	10080 minutes
monitor_path	the directory where the asnmon command was invoked.
runonce	N
trace_limit	10080 minutes

Related concepts:

- "Descriptions of the parameters that are used to operate the Replication Alert Monitor" on page 158
- "The Replication Alert Monitor" on page 143

Related tasks:

- "Operating the Replication Alert Monitor" on page 153

Descriptions of the parameters that are used to operate the Replication Alert Monitor

This topic describes the following parameters you can use to operate the Replication Alert Monitor:

- "alert_prune_limit"
- "autoprune"
- "email_server"
- "max_notification_minutes"
- "max_notifications_per_alert" on page 159
- "monitor_errors" on page 159
- "monitor_interval" on page 159
- "monitor_limit" on page 159
- "monitor_path" on page 159
- "runonce" on page 159
- "trace_limit" on page 160

alert_prune_limit

Default: alert_prune_limit=10080 minutes (seven days.)

When the Replication Alert Monitor starts a new monitor cycle, it prunes the rows from the IBMSNAP_ALERTS table that are eligible for pruning. By default, the Replication Alert Monitor prunes the rows that are older than 10080 minutes (seven days). The **alert_prune_limit** parameter controls how much old data the Replication Alert Monitor stores in the table. The parameter specifies how old the data must be before the Replication Alert Monitor prunes it.

You can reduce the value of **alert_prune_limit** parameter if the storage space on your system is small for the IBMSNAP_ALERTS table. A lower prune limit saves space, but increases processing costs. Alternatively, you might want to increase the value for the **alert_prune_limit** parameter to maintain a history of all the alert activity. In SQL replication only, a higher prune limit requires more space for change-data (CD) tables and UOW tables, but decreases processing costs.

autoprune

Default: autoprune=y

The **autoprune** parameter controls automatic pruning. The Replication Alert Monitor automatically prunes rows from the IBMSNAP_ALERTS table that it has already copied into the Monitor control tables.

email_server

The **email_server** parameter enables the ASNMAIL exit routine. The default ASNMAIL routine enables the Replication Alert Monitor to send alerts via e-mail. Set the value of this parameter to the address of an e-mail server that is set to use the Simple Mail Transfer Protocol (SMTP).

max_notification_minutes

Default: max_notifications_minutes=60

The **max_notifications_minutes** parameter specifies how long that a monitor will track an alert condition to see if it occurs more than once. By default, if an alert condition occurs more than once within 60 minutes, the Replication Alert Monitor

will send a maximum of three alerts during the 60 minute period. The **max_notifications_per_alert** parameter tells the Monitor how many notifications to send during the period of time specified by the **max_notifications_minutes** parameter for any alert condition.

max_notifications_per_alert

Default: max_notifications_per_alert=3

The **max_notifications_per_alert** parameter tells the Replication Alert Monitor the maximum number of notifications to send for any one alert. By default, if the Replication Alert Monitor receives an alert condition more than once, it sends a maximum of three notifications for that alert condition in a period of 60 minutes.

monitor_errors

The Replication Alert monitor stores any errors that occur in the monitoring process. One example of an operational error is when the Replication Alert Monitor cannot connect to the Monitor control server. You must specify an e-mail address for the **monitor_errors** parameter if you want to receive notification of operational errors. If you do not specify an e-mail address, the Replication Alert Monitor logs operational errors, but it does not send notification of the errors.

The Replication Alert Monitor ignores the **monitor_errors** parameter if the **email_server** parameter does not describe a valid e-mail server.

monitor_interval

Default: monitor_interval=300 seconds (5 minutes)

The **monitor_interval** parameter tells the Replication Alert Monitor how often to check for alert conditions. By default, the Replication Alert Monitor checks for all alert conditions for the specific monitor on the server every 300 seconds.

monitor_limit

Default: monitor_limit=10080 minutes (7 days)

For Q replication, the **monitor_limit** parameter specifies how long to keep rows in the IBMQREP_CAPMON and the IBMQREP_CAPQMON tables before the Q Capture program can prune them. For SQL replication, the **monitor_limit** parameter specifies how long to keeps rows in the IBMSNAP_CAPMON table before the Q Capture program can prune them. At each pruning interval, the Capture and Q Capture programs prune rows in these tables if the rows are older than this limit based on the current timestamp.

monitor_path

Default: monitor_path=the directory where the **asnmon** command was invoked

The **monitor_path** parameter specifies the location of the log files that the Replication Alert Monitor uses.

runonce

Default: runonce=n

When you start the Replication Alert Monitor, by default, it runs at intervals to monitor any alert conditions that you selected. You can schedule the Replication Alert Monitor to run hourly, at some other time interval, or even just one time.

When you specify **runonce**=y, the Replication Alert Monitor checks one time for all the alert conditions that you selected and ignores the **monitor_interval** parameter. You can use **runonce** when you run the Replication Alert Monitor in a batch process. For example, after the Apply program completes, you can use **runonce**=y to determine if any subscription sets failed. Then, if a subscription set did fail, the Replication Alert Monitor sends notification to your contact person or group.

By default, the **monitor_interval** is 300 seconds (five minutes). The Replication Alert Monitor checks for all the alert conditions for each monitor on the server every 300 seconds. If the Replication Alert Monitor finds an alert condition, it sends notification.

trace_limit

Default: trace_limit=10080 minutes (7 days)

The **trace_limit** parameter tells the Replication Alert Monitor how often to prune the IBMSNAP_MONTRACE and IBMSNAP_MONTRAIL tables. The Replication Alert Monitor stores the rows in these tables for 10080 minutes (seven days).The Replication Alert Monitor prunes any rows older than the value that you specify for the **trace_limit** parameter.

Related concepts:

- "Default values of the parameters that are used to operate the Replication Alert Monitor" on page 157
- "The Replication Alert Monitor" on page 143

Related tasks:

- "Operating the Replication Alert Monitor" on page 153

Setting parameters for the Replication Alert Monitor

Setting parameters for the Replication Alert Monitor—Overview

You can determine the behavior of the Replication Alert Monitor by setting the values for various parameters. You can set parameters every time you create a monitor.

Procedure:

To set parameters for the Replication Alert Monitor:

1. Specify how often the Replication Alert Monitor runs. See "Specifying how often the Replication Alert Monitor runs" on page 161 for details.
2. Specify prune intervals for data from the Replication Alert Monitor. See "Specifying prune intervals for data from the Replication Alert Monitor" on page 161 for details.
3. Specify notification criteria for selected alert conditions. See "Specifying notification criteria for selected alert conditions" on page 162 for details.
4. Specify notification criteria for operational errors. See "Specifying notification criteria for operational errors" on page 162 for details.

"E-mail notifications for replication alert conditions" on page 149 describes e-mail notification, groups, and contacts in detail.

Related tasks:

- "Setting up the Replication Alert Monitor" on page 151

Specifying how often the Replication Alert Monitor runs

You must decide how often the Replication Alert Monitor will check for alert conditions for your replication environment.

Procedure:

To specify how often the Replication Alert Monitor runs, use the following methods:

- Use the **runonce** parameter of the **asnmon** command to specify if the Replication Alert Monitor will run repeatedly or only once.
- Use the **monitor_interval** parameter of the **asnmon** command to specify how often the Replication Alert Monitor will run when **runonce**=n.
- Use the Replication Center to specify run times when you start the Replication Alert Monitor.

Related concepts:

- "Default values of the parameters that are used to operate the Replication Alert Monitor" on page 157
- "Descriptions of the parameters that are used to operate the Replication Alert Monitor" on page 158
- "The Replication Alert Monitor" on page 143

Specifying prune intervals for data from the Replication Alert Monitor

The Replication Alert Monitor can automatically prune your Monitor tables. You must decide whether the Monitor will prune the Monitor tables automatically, and if so, how the Monitor will prune the table.

Procedure:

To specify how often to prune your monitor tables, use the following methods:

- Specify whether you want the Replication Alert Monitor to automatically prune its control tables by using the **autoprune** parameter.
- Change the value for the **alert_prune_limit** parameter to control how much historic data you want the Replication Alert Monitor to store in the table. Specify how old the data must be before the Replication Alert Monitor prunes it from the IBMSNAP_ALERTS table.
- Change the value for the **trace_limit** parameter to control how long the Replication Alert Monitor stores rows in your monitor tables.

Related concepts:

- "Default values of the parameters that are used to operate the Replication Alert Monitor" on page 157
- "Descriptions of the parameters that are used to operate the Replication Alert Monitor" on page 158
- "The Replication Alert Monitor" on page 143

Specifying notification criteria for selected alert conditions

The Replication Alert monitor stores any alert conditions that you select. You can set up notification parameters to notify a contact of the alert conditions automatically via electronic mail (e-mail).

Procedure:

To specify notification criteria for alert conditions, use the following methods:

1. Set the **max_notifications_per_alert** parameter to control the maximum notification for a particular time period. Specify the maximum number of notifications you want to receive about a particular alert condition within the time period specified by the **max_notifications_minutes** parameter.
2. Set the **email_server** parameter to enable DB2 to notify you by e-mail when an alert condition occurs. Set the value of this parameter to the address of an e-mail server by using the SMTP protocol.
3. Optional: Write your own extensions to the ASNMAIL exit routine to customize how alert conditions are handled. This option is useful for integrating with problem management and other systems.

Related concepts:

- "E-mail notifications for replication alert conditions" on page 149
- "Default values of the parameters that are used to operate the Replication Alert Monitor" on page 157
- "Descriptions of the parameters that are used to operate the Replication Alert Monitor" on page 158
- "Alert conditions for the Replication Alert Monitor" on page 146
- "The Replication Alert Monitor" on page 143

Related tasks:

- "Defining contact information for the Replication Alert Monitor" on page 152

Specifying notification criteria for operational errors

The Replication Alert Monitor sends notification if it causes an error during its operation.

Procedure:

To specify notification criteria for operational errors, use the following method:

Set the value of the **monitor_errors** parameter to an e-mail address. The monitor will send notification of operational errors that it causes to this address. Enter the e-mail address by using the Simple Mail Transfer Protocol (SMTP) protocol.

Related concepts:

- "Default values of the parameters that are used to operate the Replication Alert Monitor" on page 157
- "Descriptions of the parameters that are used to operate the Replication Alert Monitor" on page 158

Stopping a monitor

When you stop a monitor, it stops checking the replication or publishing programs for alert conditions. You can use the Replication Center, a system command, or a DB2 replication service to stop a monitor.

Procedure:

To stop a monitor, use one of the following methods:

Replication Center
Use the Stop Monitor window to stop a monitor. To open the window, right-click the Monitor qualifier that identifies the monitor that you want to stop, and select **Stop Monitor**. See the online help for details.

asnmcmd system command
You can use the **asnmcmd stop** command to stop a monitor. See "asnmcmd: Working with a running Replication Alert Monitor" on page 292 for details.

Windows Service Control Manager
When you stop a DB2 replication service, the monitor will stop automatically when the replication service stops.

If the monitor stops while the Capture, Apply, Q Capture, or Q Apply programs are running, then the next time the monitor starts it performs the following actions:

- Checks for alert conditions that were met while the monitor was stopped.
- Issues alerts for any conditions that were met.

Related tasks:

- "Starting monitors" on page 156
- "Reinitializing monitors" on page 156

Chapter 12. On-demand reporting for SQL replication

This chapter describes methods you can use to report on and analyze your replication environment. Using the information in this chapter, you can check the current status of the replication programs or review historical data to determine recent messages and throughput or latency statistics.

This chapter contains the following sections:

- "Checking for current status of replication programs (Linux, UNIX, Windows, z/OS)"
- "Checking the status of the Capture and Apply journal jobs (OS/400)" on page 166
- "Reviewing historical data for trends" on page 167
- "Monitoring the progress of the Capture program (OS/400)" on page 171

Checking for current status of replication programs (Linux, UNIX, Windows, z/OS)

You can quickly assess the current status of the Capture program, Apply program, or Replication Alert Monitor program.

Use one of the following methods to check the current status of the replication programs:

Replication Center (UNIX, Windows, z/OS)

Use the Query Status window to check the current status of the Capture or Apply programs. (You cannot query the status of the Replication Alert Monitor using the Replication Center.) See the Replication Center help for details.

Command line (UNIX, Windows, z/OS)

- Capture program **asnccmd** system command, **status** parameter. See "asnccmd: Operating Capture" on page 288 for details.
- Apply program **asnacmd** system command, **status** parameter. See "asnacmd: Operating Apply" on page 271 for details.
- Replication Alert Monitor **asnmcmd** system command, **status** parameter. See "asnmcmd: Working with a running Replication Alert Monitor" on page 292 for details.

When you query the status of a program, you receive messages that describe the state of each thread that is associated with that program:

- The Capture program has four threads: Administration thread, pruning thread, worker thread, serialization thread.
- The Apply program has two threads: Administration thread, worker thread.
- The Replication Alert Monitor program has three threads: Administration thread, worker thread, serialization thread.

You can discern whether your programs are working correctly by the messages you receive. Typically worker threads, administration threads, and pruning threads are in a working state and are performing the tasks that they were intended to perform. Serialization threads are typically in the waiting state; they are global

signal handlers and are usually waiting for signals. The pruning thread prunes the CD tables and the following replication control tables.

- Unit-of-work (IBMSNAP_UOW) table
- Capture trace (IBMSNAP_CAPTRACE) table
- Capture monitor (IBMSNAP_CAPMON) table
- Signal (IBMSNAP_SIGNAL) table

If the messages that you receive indicate that the program is working but you find evidence in your environment to the contrary, you must do more investigating. For example, if you query the status of the Apply program and you find that the worker thread is working but you notice that data is not being applied to the target tables as you expected, look in the Apply trail (IBMSNAP_APPLYTRAIL) table for messages that might explain why the data is not being applied. Perhaps there are some system resource problems preventing the program from working.

If the messages you receive do not indicate a typical state, you might have to take further action as described in Table 16.

Table 16. Suggested actions for problems related to status of processing threads

Status of processing thread	Description and suggested action
Exists	The thread exists but cannot start. Contact IBM Software Support.
Was started	Investigate potential system resource problems, such as not enough CPU.
Is initializing	The thread is initialized but cannot work. Contact IBM Software Support.
Is resting	This state pertains only to threads for the Capture program. If your threads are in this state, you have suspended the Capture program and it is waiting for you to resume its operations.
Is stopped	The thread is not running. Check the Apply trail (IBMSNAP_APPLYTRAIL) or Capture trace (IBMSNAP_CAPTRACE) table for messages that explain why the thread stopped. For example, if you get a message indicating that the pruning thread stopped, check the IBMSNAP_CAPTRACE table to find out why. If your tables are too big and you want to prune them now, you can stop the Capture program and start it again to start the pruning thread.

Checking the status of the Capture and Apply journal jobs (OS/400)

On DB2 for iSeries, use the Work with Subsystem Jobs (**WRKSBSJOB**) system command to check the status of the journal jobs for the Capture and Apply programs.

1. Enter the command:

   ```
   WRKSBSJOB subsystem
   ```

   where *subsystem* is the subsystem name. In most cases the subsystem is QZSNDPR, unless you created your own subsystem description.
2. In the list of running jobs, look for the jobs you're interested in. The journal job is named after the journal to which it was assigned. If a job is not there, use the Work with Submitted Jobs (**WRKSBMJOB**) system command or the Work with

Job (**WRKJOB**) system command to locate the job. Find the job's joblog to verify that it completed successfully or to verify why it failed.

Reviewing historical data for trends

You can review historical data from your recent replication operations and evaluate the data for trends. The trends that you recognize over time might demonstrate to you that a steady volume of data is being replicated or might indicate that adjustments could be made to improve performance.

Historical data is derived from the following control tables: Apply trail (IBMSNAP_APPLYTRAIL), Apply trace (IBMSNAP_APPLYTRACE), Capture monitor (IBMSNAP_CAPMON), and Capture trace (IBMSNAP_CAPTRACE). The frequency with which you prune these tables affects the reports that you can generate. It is recommended that you keep at least one week of data in these tables so that you can examine that data when you are troubleshooting or evaluating performance.

Table 17 describes the historical data that you can view.

Table 17. Where to find historical information

To answer this question:	Use the following Replication Center window:
What are the recent messages from the Capture, Apply, and Monitor programs?	Capture Messages Apply Messages Monitor Message
On average: • How many rows were processed in the CD table for a given period of time? • How many rows were pruned? • How many transactions were committed? • How much memory is the Capture program using?	Capture Throughput Analysis
On average, what is the approximate length of time between when data was updated at the source and when it was captured by the Capture program?	Capture Latency
What are the recent messages from the Apply program?	Apply Report
On average, • How many rows were processed in the target table for a given period of time? • What is the elapsed time for processing of subscription sets?	Apply Throughput Analysis
On average, approximately how much time has elapsed between the time when the source table was updated and when the corresponding target table was updated?	End-to-End Latency

You can select a range of time to identify how much data you want to analyze. Specify the dates and times for both the beginning and the end of a time range, and then specify that the results be displayed as an average of the calculated rates. You select intervals of time (one second, one minute, one hour, one day, or one week) to group the results. For example, if you chose to analyze the Apply throughput from 9:00 p.m. and 9:59 p.m., and you want the data displayed in one minute intervals, the results are displayed in 60 rows, each row summarizing the activity for a single minute out of the 60-minute range. Alternatively, if you chose a one-hour interval, the results are displayed in 1 row, showing the average throughput for the specified hour time period. If you don't specify an interval, you can view raw data from the APPLYTRAIL table.

The Replication Center windows show results from information contained in various control tables and log files. The following sections describe in more detail how you can use historical data to evaluate your replication operations with the Replication Center:

- "Reviewing Capture program messages"
- "Examining Capture program throughput"
- "Displaying latency of data processed by the Capture program" on page 169
- "Reviewing Apply program messages" on page 169
- "Examining Apply program throughput" on page 170
- "Displaying the average length of time taken to replicate transactions" on page 170
- "Reviewing Monitor program messages" on page 171

Reviewing Capture program messages

Use the Capture Messages window to review the messages that were inserted in the Capture trace (IBMSNAP_CAPTRACE) table over a specified period of time. The IBMSNAP_CAPTRACE table contains a row for significant events such as initialization, pruning, warnings, and errors that are issued by the Capture program.

For example, from the Capture Messages window, you can review all the error and warning messages that are recorded by the Capture program during one week. You can also print or save data to a file from the Capture Messages window.

Examining Capture program throughput

Use the Capture Throughput Analysis window to display the performance results of a Capture program over a specific range of time. The Capture program regularly records statistical information in the Capture monitor (IBMSNAP_CAPMON) table, and during pruning it records pruning statistics in the Capture trace (IBMSNAP_CAPTRACE) table. Using information from these tables, the Capture Throughput Analysis window displays the calculated results of the performance rates of four different tasks.

You can examine the results of all four types of information to assess the throughput performance of the Capture program. You can specify that the results are displayed in absolute or average values.

- Number of rows inserted from log or skipped
- Number of rows pruned from CD table
- Number of transactions committed
- Use of memory

For example, from the Capture Throughput Analysis window, you can review the average weekly performance of the Capture program throughput. To do so, specify the dates and times for both the beginning and the end of a time range, and then specify that the results be displayed as an average of the calculated rates.

Displaying latency of data processed by the Capture program

Use the Capture Latency window to display the approximate length of time between when data was updated at the source and when it was captured by the Capture program. The elapsed time gives you an indication of the currency of the data in those CD tables over time. This average latency is derived from information the Capture monitor (CAPMON) table, which derives its information from the register (IBMSNAP_REGISTER) table.

The current Capture latency is calculated using the CURRENT_TIMESTAMP value in the SYNCHTIME column from the global record in the register (IBMSNAP_REGISTER) table:

```
(CURRENT_TIMESTAMP) - (SYNCHTIME)
```

Table 18. Example of values for calculating current Capture latency

Parameter	Column value
CURRENT_TIMESTAMP	2001–10–20–10:30:25
SYNCHTIME	2001–10–20–10:30:00

For example, using the values in Table 18, the current latency is 25 seconds:

```
10:30:25 - 10:30:00
```

The Capture latency changes as time goes by and the history of these changes is stored in the Capture monitor (IBMSNAP_CAPMON) table. The Replication Center uses the information in the Capture monitor table to calculate average or historical latency. The formula for average latency is similar to the one used for current latency; however, the MONITOR_TIME value is used instead of the CURRENT_TIMESTAMP value. The MONITOR_TIME value is a timestamp indicating when the Capture program inserted a row in the Capture monitor table. You can show the average latency per second, minute, hour, day, or week. For example, from the Capture Latency window, you can display the average latency for a Capture program per hour, over the duration of the past week.

Reviewing Apply program messages

Use the Apply Messages window to review the messages that were inserted in the Apply trace (IBMSNAP_APPLYTRACE) table over a specified period of time. The IBMSNAP_APPLYTRACE table contains rows for significant events such as initialization, warnings, and errors that are issued by the Apply program.

For example, from the Apply Messages window, you can review all the error and warning messages that are recorded by the Apply program during one week. You can also print or save data to a file from the Apply Messages window.

Use the Apply Report window to check the success of a specific Apply program over a specific period of time by reviewing the data that was inserted into the Apply trail (IBMSNAP_APPLYTRAIL) table. The IBMSNAP_APPLYTRAIL table contains data about the execution of subscription sets, and includes the status of the subscription set, error messages, and the number of rows processed.

In the Apply Report window, you can display the following data:

- All subscription sets
- Failed subscription sets
- Successful subscription sets
- Error summary per failed subscription set

For example, from the Apply Report window, you can determine whether the Apply program successfully processed subscription sets during the last week. If there were subscription sets that could not be replicated, you can view the error messages issued by the Apply program for those sets. In addition, you can use the Apply Report window with the Apply Throughput Analysis window. After you use the Apply Report window to find out which sets were successfully replicated, you can use the Apply Throughput Analysis window to see the number of rows replicated and the length of time that replication took.

You can also use the Apply Report window to display all the data from a particular row in the IBMSNAP_APPLYTRAIL table.

Examining Apply program throughput

Use the Apply Throughput Analysis window to examine the performance statistics for a specific Apply qualifier. You can filter and group data without writing SQL statements. For example, you can view the number of rows that were inserted into, updated in, deleted from, and reworked in the target tables in the subscription set that was processed by a specific Apply qualifier. You can also view the time the Apply program spent processing subscription sets for a particular Apply qualifier.

Displaying the average length of time taken to replicate transactions

Use the End-to-End Latency window to display an approximate value for the average length of time used to replicate transactions in a particular subscription set. See the Replication Center help for a description of the sequence of events that take place in change-capture replication.

From the End-to-End Latency window, for example, you can view the approximate latency for a subscription set for each Apply cycle during a range of time. You can also divide the time into intervals and display the average latency for each interval.

The Replication Center uses the following formula to calculate the end-to-end latency:

```
(ENDTIME - LASTRUN) + (SOURCE_CONN_TIME - SYNCHTIME)
```

where:

- ENDTIME is the time at which the Apply program finishes processing the subscription set.
- LASTRUN is the time at which the Apply program starts processing a subscription set.
- SOURCE_CONN_TIME is the time at which the Apply program connects to the Capture control server to fetch data.
- SYNCHTIME is the time of the most current commit of data to the CD tables by the Capture program.

Table 19. Example of values for calculating end-to-end latency

Parameter	Column value
ENDTIME	2001–10–20–10:01:00
LASTRUN	2001–10–20–10:00:30
SOURCE_CONN_TIME	2001–10–20–10:00:32
SYNCHTIME	2001–10–20–10:00:00

For example, assume a particular subscription set has the values that are shown in Table 19. Using the previous equation, the average end-to-end latency for this subscription set is 62 seconds:

```
(10:01:00 - 10:00:30) + (10:00:32 - 10:00:00) = 62
```

Reviewing Monitor program messages

Use the Monitor Messages window to review the messages that were inserted in the Monitor trace (IBMSNAP_MONTRACE) table over a specified period of time. The IBMSNAP_MONTRACE table contains rows for significant events such as actions, warnings, and errors that are issued by the Monitor program.

For example, from the Monitor Messages window, you can review all the error and warning messages that are recorded by the Monitor program during one week. You can also print or save data to a file from the Monitor Messages window.

Monitoring the progress of the Capture program (OS/400)

If the Capture program has terminated, you can inspect the restart table (IBMSNAP_RESTART) to determine how much progress the Capture program made. There is one row for each journal used by the source tables. The LOGMARKER column provides the timestamp of the last journal entry processed successfully. The SEQNBR column provides the journal entry sequence number of that entry.

If the Capture program is still running, you can determine its progress using the following steps:

1. For each source table being captured, open its CD table.
2. In the last row of the CD table, note the hex value in the COMMITSEQ column.
3. Look in the Unit-of-work (IBMSNAP_UOW) table for a row with the same COMMITSEQ value. If no matching COMMITSEQ exists in the IBMSNAP_UOW table, repeat the same process with the second-to-last row in the CD table. Proceed backward through the CD table until you find a match.
4. When you find a matching COMMITSEQ, note the value in the LOGMARKER column of the UOW row. This is the timestamp of the processed journal entry. All changes to the source table up to that time are ready to be applied.
5. Use the Display Journal (**DSPJRN**) system command to determine how many journal entries remain to be processed by the Capture program. Direct the output to an output file (or to a printer for a printed report), as shown in the following example:

```
DSPJRN FILE(JRNLIB/DJRN1)
       RCVRNG(*CURCHAIN)
       FROMTIME(timestamp)
       TOTIME(*LAST)
```

```
JRNCDE(J F R C)
OUTPUT(*OUTFILE)
ENTDTALEN(1) OUTFILE(library/outfile)
```

where *timestamp* is the timestamp that you identified in 4 on page 171.

The number of records in the output file is the approximate number of journal entries that remain to be processed by the Capture program.

Chapter 13. Making changes to an SQL replication environment

This chapter covers the issues that you will need to consider when you make day-to-day changes to your replication environment.

This chapter contains the following sections:

- "Registering new objects"
- "Changing registration attributes for registered objects" on page 174
- "Adding columns to source tables" on page 174
- "Stop capturing changes for registered objects" on page 177
- "Reactivating registrations" on page 178
- "Removing registrations" on page 179
- "Changing Capture schemas" on page 179
- "Creating new subscription sets" on page 182
- "Adding new subscription-set members to existing subscription sets" on page 182
- "Disabling subscription-set members from existing subscription sets" on page 183
- "Enabling subscription-set members to existing subscription sets" on page 183
- "Changing attributes of subscription sets" on page 183
- "Changing subscription set names" on page 184
- "Splitting a subscription set" on page 186
- "Merging subscription sets" on page 189
- "Changing Apply qualifiers of subscription sets" on page 192
- "Deactivating subscription sets" on page 194
- "Removing subscription sets" on page 195
- "Coordinating replication events with database application events" on page 196
- "Promoting your replication configuration to another system" on page 203

Registering new objects

You can register a new table, view, or nickname in your replication environment at any time. You do not need to reinitialize the Capture program.

Procedure:

Use one of the following methods to register an object:

Replication Center
Use the Register Tables, Register Views, or Register Nicknames window. See the Replication Center help for details.

ADDDPRREG system command (OS/400)
See "ADDDPRREG: Adding a DPR registration (OS/400)" on page 319 for parameter descriptions and command syntax.

A new registered object is automatically initialized by the Capture program the first time that the Apply program processes a subscription set that refers to that object. The Apply program signals the Capture program to begin capturing changes for this new object. See Chapter 3, “Registering tables and views as SQL replication sources,” on page 35 for more information about registering objects.

Changing registration attributes for registered objects

You can change the registration attributes of existing registered objects at any time. These registration attributes include:

- CHGONLY
- CONFLICT_LEVEL
- RECAPTURE
- DISABLE_REFRESH
- CHG_UPD_TO_DEL_INS
- STOP_ON_ERROR
- BEFORE_IMG_PREFIX

Note: You can update the before-image prefix value only if this value is null.

Procedure:

1. Use the following method to change the attributes:

   Replication Center
   From the Registered Tables folder, right-click the registered table in the contents pane and select Properties. See the Replication Center help for details.

2. After you change the attributes, you must reinitialize the Capture program in order for it to recognize the changes. Reinitialize the Capture program by using one of the following methods:

   Replication Center
   From the Capture Control Servers folder, right-click the Capture control server in the contents pane and select Reinitialize Capture. See the Replication Center help for details.

   asnccmd system command (Windows, UNIX, z/OS)
   Use the **reinit** parameter. See “asnccmd: Operating Capture” on page 288 for parameter descriptions and command syntax.

   INZDPRCAP system command (OS/400)
   See “INZDPRCAP: Reinitializing DPR Capture (OS/400)” on page 374 for parameter descriptions and command syntax.

Adding columns to source tables

If you need to add columns to a registered source table, first consider how DB2 replication uses this table. If you need to replicate the new columns in this source table, you must ensure that the existing Capture and Apply programs recognize the new columns and continue processing without interruption. You might need to perform special processing steps depending on whether or not you want to replicate the data in the new columns.

Not replicated
If you do not want to replicate the data in the new columns, you do not need to perform any special processing steps. The Capture program immediately recognizes the changes and continues running.

Replicated

If you want to replicate the data in these new columns, follow these steps to ensure that the new column data is captured and that the Capture and Apply programs continue to run without errors.

Prerequisite:

Before using this procedure, familiarize yourself with the structures of your source, change-data (CD), and target tables and with the registrations and subscription sets defined on your system.

Restrictions:

Do not use these steps if you are adding columns to an iSeries table that uses a relative record number (RRN) as the primary key. The RRN must be the last column in the CD table. When adding columns to an iSeries table with an RRN, remove the registration, add the column to the source table, and then add this table again as a new registration specifying that the RRN will be captured. See "RMVDPRREG: Removing a DPR registration (OS/400)" on page 380 and "ADDDPRREG: Adding a DPR registration (OS/400)" on page 319 for more information about removing and adding iSeries registrations.

You cannot use these steps to add columns to registered sources on non-DB2 relational databases. A registration for a non-DB2 relational source includes a set of triggers used for capturing changes. You cannot alter these triggers. Therefore, if you need to add new columns to this source table and need to replicate the data in these columns, you must drop and re-create the existing registered source.

Procedure:

1. Quiesce all activity against the source table that you want to alter.
2. Use one of the following methods to stop the Capture program:

   Replication Center
   Use the Stop Capture window. See the Replication Center help for details.

   asnccmd system command (Windows, UNIX, z/OS)
   Use the **stop** parameter. See "asnccmd: Operating Capture" on page 288 for parameter descriptions and command syntax.

   ENDDPRCAP system command (OS/400)
   See "ENDDPRCAP: Stopping Capture (OS/400)" on page 364 for parameter descriptions and command syntax.

   Tip: If you need to keep the Capture program active during this procedure, insert a USER signal in the signal (IBMSNAP_SIGNAL) table after stopping activity against the source table. Wait for the Capture program to process the USER signal.

   After the Capture program processes the USER signal, the Capture program has no more activity to process against the associated CD table and no longer requires access to this CD table.
3. Use the following method to deactivate all subscription sets that subscribe to this source table:

   Replication Center
   From the Subscription Sets folder, right-click the active subscription sets in the contents pane and select Deactivate. See the Replication Center help for details.

Note: If you do not want to deactivate the subscription sets during this process, verify that no Apply programs associated with these subscriptions sets will be running against this source table when you are adding the new columns. Alternatively, ensure that these Apply programs have processed data up to the signal log sequence number (LSN) that is associated with the prior USER signal.

The methods in this step ensure exclusive access to the CD table so that you can alter the table.

4. Use SQL to submit an ALTER TABLE ADD statement to add the new columns to the source table.
5. Use the following method to add the new columns to the CD table:

   Replication Center
   From the Registered Tables folder, right-click the registered table in the contents pane and select Properties. See the Replication Center help for details.

   The Capture program automatically reinitializes the registration and captures the changes to these new columns when the Capture program first reads log data with the new columns.
6. Use SQL to submit an ALTER TABLE ADD statement to add the new columns to the target table.
7. Use the following method to deactivate any associated subscription sets that you did not already deactivate in step 3:

   Replication Center
   From the Subscription Sets folder, right-click the active subscription sets in the contents pane and select Deactivate. See the Replication Center help for details.

   If absolutely necessary, you can now resume activity against this source table. However, because the associated subscriptions sets have not yet been changed, you must keep these subscription sets deactivated so that you do not lose any changes made to these new columns.
8. Use the following method to add the new columns to the associated subscription-set members:

   Replication Center
   Use the Add Column to Target Table window. See the Replication Center help on adding columns to target tables for details.
9. **For UNIX, Windows, z/OS:** If you are running the Apply program with **opt4one** set to y, stop and then restart the Apply program.
10. Use the following method to reactivate the subscription sets:

    Replication Center
    From the Subscription Sets folder, right-click the deactivated subscription sets in the contents pane and select Activate. See the Replication Center help for details.

Stop capturing changes for registered objects

You should deactivate a registered object before you delete it to ensure that the Capture programs finish any necessary processing of the object. Also, you can deactivate a registered object if you want to stop capturing changes for this object temporarily but need to keep your Capture programs running for other registered objects.

The Capture program stops capturing changes for the source objects that have been deactivated; however, the change-data (CD) tables, registration attributes, and subscription sets that are associated with these source objects remain on the system.

Before you deactivate a registered object, you should deactivate all of the subscriptions sets that are associated with this registered object. This ensures that your Apply programs will not interfere with the deactivation process by automatically reactivating the object before you delete it or before you are ready to reactivate it.

All subscription sets that are associated with the registered object are affected when the object is deactivated and when DB2 replication stops capturing changes for that object. If you want to continue running these subscription sets, you must remove the subscription-set members that use this registered object as a source from the deactivated subscription sets.

Restrictions:

You can deactivate only DB2 registered objects that are defined as Capture program sources.

You cannot deactivate non-DB2 relational database objects that are used by Capture triggers.

Procedure:

To deactivate a registered object:

1. Deactivate all associated subscription sets using the following method:

   Replication Center
   From the Subscription Sets folder, right-click the active subscription sets in the contents pane and select Deactivate. See the Replication Center help for details.

   See "Deactivating subscription sets" on page 194 for more information.
2. Deactivate the registered object using one of the following methods:

   Replication Center
   From the Registered Tables folder, right-click the registered table in the contents pane and select Stop Capturing Changes. See the Replication Center help for details.

   CAPSTOP signal
   Manually insert a CAPSTOP signal in the signal (IBMSNAP_SIGNAL) table. See "*schema*.IBMSNAP_SIGNAL" on page 458 for more information.

Reactivating registrations

If you temporarily deactivate your registration and associated subscription sets and then want to reactivate the registration to begin recapturing data, you would simply reactivate these subscription sets through the Replication Center. The Capture program reactivates the registration after the Apply program sends a CAPSTART signal.

If, however, the Capture program deactivates a registration because of an unexpected error, you must take special action to reactivate the registration. Unexpected errors cause the Capture program to set the value of the STATE column to S (Stopped) in the register (IBMSNAP_REGISTER) table if the STOP_ON_ERROR column value for this registration is set to N. This STATE column value indicates that the Capture program stopped processing this registration and that the registration must be repaired. The Apply program does not issue a CAPSTART signal for any registration that is in a stopped state.

Use the following procedure to correct these unexpected errors and to make the registration eligible for reactivation.

Prerequisites:

Read the error messages that were generated by the Capture program regarding this deactivated registration.

Familiarize yourself with the structure of the DB2 replication Capture control tables and with the Capture programs running on your system.

Procedure:

1. Change your registration by using the information contained in the error messages.
2. From the Capture control server, run the following SQL script to reset the STATE column in the IBMSNAP_REGISTER table:

```
UPDATE Schema.IBMSNAP_REGISTER
   SET STATE             = 'I'
 WHERE
       SOURCE_OWNER      = 'SrcSchema'  AND
       SOURCE_TABLE      = 'SrcTbl'     AND
       SOURCE_VIEW_QUAL  =  SrcVwQual   AND
       STATE             = 'S';
```

   where *Schema* is the name of the Capture schema, *SrcSchema* is the registered source table schema, *SrcTbl* is the name of the registered source table, and *SrcVwQual* is the source-view qualifier for this source table.

After the STATE column is set to I (Inactive), the Capture program is ready to begin capturing data as soon as a CAPSTART signal is received, usually from the Apply program.

Example: Suppose that the source table for an active registration was inadvertently altered to DATA CAPTURE NONE (and should be DATA CAPTURE CHANGES). Also, suppose that this registration was defined with STOP_ON_ERROR = 'N', which specifies that the Capture program will not stop when it encounters errors. At the next restart or reinitialization of the Capture program, the Capture program will recognize this incorrect condition of the source table and will set the STATE column to S (Stopped) in the register (IBMSNAP_REGISTER) table for this

registration. You will receive an error message when the Apply program tries to process the corresponding subscription set, because the registration will be in a stopped state. You must:

- Correct the setting of the source table through SQL by submitting an ALTER TABLE statement that resets the table option to DATA CAPTURE CHANGES.
- Manually reset the registration from a stopped state to an inactive state, using the above SQL script.

The Apply program will then perform a full refresh of the entire subscription set.

Removing registrations

If you remove a registration, DB2 replication removes the registration of the object, drops the associated change-data (CD) or consistent-change data (CCD) tables, and drops the CCD object nickname and any Capture triggers for non-DB2 relational database sources. The actual source table or view remains in the database.

Prerequisite:

Deactivate the source object first to ensure that the Capture program finishes any current processing of this object.

Important: Deactivation is an asynchronous process. Be sure that the deactivation process finishes before you remove the object.

Procedure:

Use one of the following methods to remove a registration for a source table or view:

Replication Center
Use the Delete Registered Tables or Delete Registered Views window. See the Replication Center help for details.

RMVDPRREG system command (OS/400)
See "RMVDPRREG: Removing a DPR registration (OS/400)" on page 380 for parameter descriptions and command syntax.

Changing Capture schemas

You can use the following procedure to change an existing Capture schema.

Prerequisites:

Before running the following SQL statements, familiarize yourself with your DB2 replication control tables and with the subscription sets that are defined on your system.

For UNIX, Windows, z/OS: If you set up monitoring definitions or started Replication Alert Monitor programs under the Capture schema that you are going to change, drop these monitoring definitions. After you change the Capture schema, re-create the monitoring definitions with the new Capture schema name through the Replication Center. Then, you can reinitialize the associated Replication Alert Monitor programs using the **asnmcmd** system command with the **reinit** parameter. Alternatively, you can stop the Replication Alert Monitor

programs using the **asnmcmd** system command with the **stop** parameter and then restart the programs using the **asnmon** system command.

Determine the new Capture schema name. See Chapter 17, “Naming rules for SQL replication objects,” on page 269 for more information.

Verify that your Capture control server and all of the Apply control servers that are associated with this Capture control server have been migrated to Version 8 before you use this procedure.

Restriction:

You should not use this procedure if your source server is a non-DB2 relational database.

Procedure:

1. Use one of the following methods to create control tables for a new Capture schema:

   Replication Center (UNIX, Windows, z/OS)
   Use the Create Replication Control Tables notebook. See the Replication Center help for details.

   CRTDPRTBL system command (OS/400)
   See “CRTDPRTBL: Creating the replication control tables (OS/400)” on page 360 for parameter descriptions and command syntax.

2. Use one of the following methods to stop the Capture program that is using the existing Capture schema. (Skip this step if you do not have a Capture program running.):

   Replication Center
   Use the Stop Capture window. See the Replication Center help for details.

   asnccmd system command (UNIX, Windows, z/OS)
   Use the **stop** parameter. See “asnccmd: Operating Capture” on page 288 for parameter descriptions and command syntax.

   ENDDPRCAP system command (OS/400)
   See “ENDDPRCAP: Stopping Capture (OS/400)” on page 364 for parameter descriptions and command syntax.

3. Use the following method to deactivate all associated subscription sets:

   Replication Center
   From the Subscription Sets folder, right-click the active subscription sets in the contents pane and select Deactivate. See the Replication Center help for details.

4. From the Apply control server, run the following SQL statement to change the Capture schema names for the associated subscription sets with source tables that belong to this Capture schema:

   ```
   UPDATE ASN.IBMSNAP_SUBS_SET
      SET CAPTURE_SCHEMA    = 'NewSchema'
    WHERE
          CAPTURE_SCHEMA    = 'ExistingSchema';
   ```

   where *NewSchema* is the new Capture schema name, and *ExistingSchema* is the name of the Capture schema that you are changing.

5. If you created subscription sets with target tables (for example, CCD or replica type tables) that are registered in this Capture schema, run the following SQL statement from the Apply control server to change the target schema name of these subscription sets:

```
UPDATE ASN.IBMSNAP_SUBS_SET
   SET TGT_CAPTURE_SCHEMA    = 'NewSchema'
 WHERE
       TGT_CAPTURE_SCHEMA    = 'ExistingSchema';
```

   where *NewSchema* is the new Capture schema name, and *ExistingSchema* is the name of the Capture schema that you are changing.

6. From the Capture control server, run an SQL statement to copy the active information from each existing Capture control table to each new corresponding Capture control table that you created in step 1. For example, to copy the active information to the new register (IBMSNAP_REGISTER) table:

```
INSERT INTO NewSchema.IBMSNAP_REGISTER
       SELECT * FROM
              ExistingSchema.IBMSNAP_REGISTER;
```

   where *NewSchema* is the new Capture schema name, and *ExistingSchema* is the name of the Capture schema that you are changing.

   Repeat this step for each existing Capture control table, including some or all of the following tables:

   - IBMSNAP_CAPMON
   - IBMSNAP_CAPPARMS
   - IBMSNAP_CAPTRACE
   - IBMSNAP_PRUNCNTL
   - IBMSNAP_PRUNE_SET
   - IBMSNAP_REG_EXT (OS/400 only)
   - IBMSNAP_REGISTER
   - IBMSNAP_RESTART
   - IBMSNAP_SIGNAL
   - IBMSNAP_UOW

   (You do not need to repeat this step for the IBMSNAP_CAPENQ [on UNIX, Windows, z/OS] or the IBMSNAP_PRUNE_LOCK control table, because there are no rows in these tables.)

   Do *not* change the CD tables.

7. Use the following method to drop the existing schema and its associated Capture control tables:

   Replication Center
   From the Capture Control Servers folder, right-click the database for which you want to remove Capture control tables and select Drop Capture Control Tables. See the Replication Center help for details.

8. Use one of the following methods to restart the Capture program with the new schema name:

   Replication Center
   Use the Start Capture window. See the Replication Center help for details.

   asncap system command (UNIX, Windows, z/OS)
   Use the **capture_schema**=*NewSchema* and **startmode**=warmsi or

warmns parameter options. See "asncap: Starting Capture" on page 282 for parameter descriptions and command syntax.

STRDPRCAP system command (OS/400)
Use the **RESTART(*YES)** parameter. See "STRDPRCAP: Starting Capture (OS/400)" on page 393 for parameter descriptions and command syntax.

9. Reactivate the associated subscription sets using the following method:

Replication Center
From the Subscription Sets folder, right-click the deactivated subscription sets in the contents pane and select Activate. See the Replication Center help for details.

Creating new subscription sets

You can create new subscription sets and add new subscription-set members to sets at any time for an existing registered object.

This procedure addresses the addition of a new subscription set, with or without subscription-set members.

Prerequisites:

Before you create a new subscription set, register the tables or views that you want to use as sources.

Procedure:

Use one of the following methods to create a new subscription set:

Replication Center
Use the Create Subscription Set notebook. See the Replication Center help for details.

ADDDPRSUB system command (OS/400)
See "ADDDPRSUB: Adding a DPR subscription set (OS/400)" on page 327 for parameter descriptions and command syntax.

See Chapter 4, "Subscribing to sources for SQL replication," on page 57 for additional information.

Important: If the corresponding Apply program is active, do not activate the new subscription set until the subscription set is fully defined.

Adding new subscription-set members to existing subscription sets

Procedure:

Use the Replication Center to add a new subscription-set member to an existing subscription set.

Replication Center
Use the Add Member to Subscription Set notebook. See the Replication Center help for details.

The Apply program will perform a full refresh on any new members that are added to the set on the next Apply cycle. Replication changes to all target tables will continue on subsequent cycles. If you are running the Apply program with the **opt4one** variable set to y, you must stop and restart the Apply program before the new member is processed by Apply.

Disabling subscription-set members from existing subscription sets

If there is a problem replicating to a table in the subscription set, the Apply program will place the error messages into the IBMSNAP_APPLYTRAIL table and and no members of the subscription set will be processed on this apply cycle. If you want the Apply program to ignore the failing subscription-set member and continue processing the rest of the subscription set, you must disable the failing subscription-set member. Use the following SQL UPDATE statement to disable a subscription-set member:

```
UPDATE ASN.IBMSNAP_SUBS_MEMBR
 SET MEMBER_STATE = 'D'
 WHERE APPLY_QUAL= apply_qualifier
                SET_NAME = set_name
                 WHOS_ON_FIRST = whos_on_first
                 SOURCE_OWNER = source_owner
   SOURCE_TABLE = source_table
  SOURCE_VIEW_QUAL = source_view_qualifier
  TARGET_OWNER = target_owner
  TARGET_TABLE = target_table
```

The Apply program will not process this member until the member is re-enabled.

Enabling subscription-set members to existing subscription sets

You can add or re-enable disabled members in a subscription set by changing the MEMBER_STATE to 'N' (new). Use the following SQL UPDATE statement to re-enable a subscription-set member:

```
UPDATE ASN.IBMSNAP_SUBS_MEMBR
 SET MEMBER_STATE = 'N'
 WHERE APPLY_QUAL= apply_qualifier
                SET_NAME = set_name
                 WHOS_ON_FIRST = whos_on_first
                 SOURCE_OWNER = source_owner
   SOURCE_TABLE = source_table
  SOURCE_VIEW_QUAL = source_view_qualifier
  TARGET_OWNER = target_owner
  TARGET_TABLE = target_table
```

Changing attributes of subscription sets

You might need to change an attribute of an existing subscription set. Attributes that you might need to change include:

- Schedules for applying updates (time-based replication or event-based replication)
- Subscription statements
- WHERE clause predicates of subscription-set members
- Commit count
- Data blocking value (MAX_SYNCH_MINUTES)

Procedure:

To change an attribute of a subscription set, perform the following steps through the Replication Center:

1. Deactivate the subscription set.
2. Change the subscription set and any subscription-set members.
3. Reactivate the subscription set.

By first deactivating the subscription set, you keep the Apply program up and running but prevent the Apply program from processing this subscription set while you enter your changes. The Apply program recognizes your subscription set changes during the next Apply cycle after you reactivate the subscription set.

Note: If you set the **opt4one** Apply program parameter to y, your changes are not recognized unless you stop and then restart the Apply program (UNIX, Windows, z/OS).

Changing subscription set names

Use the following procedure to change the name of a subscription set without having to drop and then re-create the subscription set and all of its members.

Prerequisites:

Before running these SQL statements, familiarize yourself with the structure of the DB2 replication control tables and with the subscription sets defined on your system.

For UNIX, Windows, z/OS: If you set up monitoring definitions or started Replication Alert Monitor programs to detect alert conditions for the subscription set that you are going to change, drop these monitoring definitions. After you change the subscription-set name, re-create the monitoring definitions through the Replication Center. Then, you can reinitialize the associated Replication Alert Monitor programs using the **asnmcmd** system command with the **reinit** parameter. Alternatively, you can stop the Replication Alert Monitor programs using the **asnmcmd** system command with the **stop** parameter and then restart the programs using the **asnmon** system command.

Determine the new subscription set name that you want to use.

Procedure:

1. Use the following method to deactivate the subscription set that you want to change:

   Replication Center
   From the Subscription Sets folder, right-click the active subscription set in the contents pane and select Deactivate. See the Replication Center help for details.

2. From the Apply control server, run the following SQL statements to change the name of the subscription set in the subscription sets (IBMSNAP_SUBS_SET), subscription members (IBMSNAP_SUBS_MEMBR), and subscription columns (IBMSNAP_SUBS_COLS) tables:

```
UPDATE ASN.IBMSNAP_SUBS_SET
   SET SET_NAME      = 'NewSetName'
 WHERE
       APPLY_QUAL    = 'ApplyQual'      AND
       SET_NAME      = 'ExistSetName'   AND
       WHOS_ON_FIRST = 'Val';
```

```
UPDATE ASN.IBMSNAP_SUBS_MEMBR
   SET SET_NAME      = 'NewSetName'
 WHERE
       APPLY_QUAL    = 'ApplyQual'      AND
       SET_NAME      = 'ExistSetName'   AND
       WHOS_ON_FIRST = 'Val';
UPDATE ASN.IBMSNAP_SUBS_COLS
   SET SET_NAME      = 'NewSetName'
 WHERE
       APPLY_QUAL    = 'ApplyQual'      AND
       SET_NAME      = 'ExistSetName'   AND
       WHOS_ON_FIRST = 'Val';
```

where *NewSetName* is the new subscription set name, *ApplyQual* is the Apply qualifier, *ExistSetName* is the existing name of the subscription set, and *Val* is either F or S.

3. If this subscription set uses before or after SQL statements or procedure calls, run the following SQL script from the Apply control server to change the subscription set name in the subscription statements (IBMSNAP_SUBS_STMTS) table:

```
UPDATE ASN.IBMSNAP_SUBS_STMTS
   SET SET_NAME      = 'NewSetName'
 WHERE
       APPLY_QUAL    = 'ApplyQual'      AND
       SET_NAME      = 'ExistSetName'   AND
       WHOS_ON_FIRST = 'Val';
```

where *NewSetName* is the new subscription set name, *ApplyQual* is the Apply qualifier, *ExistSetName* is the existing name of the subscription set, and *Val* is either F or S.

4. From the Capture control server, run the following SQL statements to change the subscription set name in the prune set (IBMSNAP_PRUNE_SET) and pruning control (IBMSNAP_PRUNCNTL) tables:

```
UPDATE Schema.IBMSNAP_PRUNE_SET
   SET SET_NAME      = 'NewSetName'
 WHERE
       APPLY_QUAL    = 'ApplyQual'     AND
       SET_NAME      = 'ExistSetName'  AND
       TARGET_SERVER = 'Target_Server';
UPDATE Schema.IBMSNAP_PRUNCNTL
   SET SET_NAME      = 'NewSetName'
 WHERE
       APPLY_QUAL    = 'ApplyQual'     AND
       SET_NAME      = 'ExistSetName'  AND
       TARGET_SERVER = 'Target_Server';
```

where *Schema* is the name of the Capture schema, *NewSetName* is the new subscription set name, *ApplyQual* is the Apply qualifier, *ExistSetName* is the existing name of the subscription set, and *Target_Server* is the database location of the target tables.

5. **For UNIX, Windows, z/OS:** If you are running the Apply program with **opt4one** set to y, stop and then restart the Apply program.
6. Reactivate the subscription set using the following method:

   Replication Center

   From the Subscription Sets folder, right-click the deactivated subscription set in the contents pane and select Activate. See the Replication Center help for details.

Splitting a subscription set

You can use the following procedure to split a subscription set into two or more subscription sets without having to remove and re-create subscription set information.

Prerequisites:

Before running these SQL statements, familiarize yourself with the structure of the DB2 replication control tables and with the subscription sets defined on your system.

Identify the subscription-set members of the subscription set that you want to split, and determine the source and target tables associated with these subscription-set members.

Identify the Capture control server, target server, and Apply control server of the subscription set that you want to split. You must use these Capture control server, target server, and Apply control server locations for the new subscription set that you want to create using this procedure.

For UNIX, Windows, z/OS: If you set up monitoring definitions or started Replication Alert Monitor programs to detect alert conditions for the subscription set that you are going to split, drop these monitoring definitions. After you split the subscription set, re-create the monitoring definitions through the Replication Center. Then, you can reinitialize the associated Replication Alert Monitor programs using the **asnmcmd** system command with the **reinit** parameter. Alternatively, you can stop the Replication Alert Monitor programs using the **asnmcmd** system command with the **stop** parameter and then restart the programs using the **asnmon** system command.

Procedure:

1. Use the following method to deactivate the subscription set that you want to split:

   Replication Center
   From the Subscription Sets folder, right-click the active subscription set in the contents pane and select Deactivate. See the Replication Center help for details.

2. Use one of the following methods to create a new subscription set:

   Replication Center
   Use the Create Subscription Set notebook. See the Replication Center help for details.

   ADDDPRSUB system command (OS/400)
   Use the **SRCTBL(*NONE)**, **TGTTBL(*NONE)**, and **ACTIVATE(*NO)** parameter options. See "ADDDPRSUB: Adding a DPR subscription set (OS/400)" on page 327 for parameter descriptions and command syntax.

   Both of these methods create a new row in the subscription sets (IBMSNAP_SUBS_SET) table.

   Leave this new subscription set inactive.

3. From the Apply control server, run the following SQL statement to copy information from the existing subscription set into the new subscription set row in the IBMSNAP_SUBS_SET table:

```
UPDATE ASN.IBMSNAP_SUBS_SET
   SET STATUS     =
                  (SELECT STATUS FROM ASN.IBMSNAP_SUBS_SET B
                    WHERE APPLY_QUAL    = 'ApplyQual' AND
                          SET_NAME      = 'ExistName' AND
                          WHOS_ON_FIRST = 'Val'),
       LASTRUN    =
                  (SELECT LASTRUN FROM ASN.IBMSNAP_SUBS_SET B
                    WHERE APPLY_QUAL    = 'ApplyQual' AND
                          SET_NAME      = 'ExistName' AND
                          WHOS_ON_FIRST = 'Val'),
       SYNCHPOINT  =
                   (SELECT SYNCHPOINT FROM ASN.IBMSNAP_SUBS_SET B
                    WHERE APPLY_QUAL    = 'ApplyQual' AND
                          SET_NAME      = 'ExistName' AND
                          WHOS_ON_FIRST = 'Val'),
       SYNCHTIME   =
                   (SELECT SYNCHTIME FROM ASN.IBMSNAP_SUBS_SET B
                    WHERE APPLY_QUAL    = 'ApplyQual' AND
                          SET_NAME      = 'ExistName' AND
                          WHOS_ON_FIRST = 'Val'),
       LASTSUCCESS =
                   (SELECT LASTSUCCESS FROM ASN.IBMSNAP_SUBS_SET B
                    WHERE APPLY_QUAL    = 'ApplyQual' AND
                          SET_NAME      = 'ExistName' AND
                          WHOS_ON_FIRST = 'Val')
 WHERE
       APPLY_QUAL         = 'ApplyQual' AND
       SET_NAME           = 'NewName'   AND
       WHOS_ON_FIRST      = 'Val';
```

where *ApplyQual* is the Apply qualifier, *ExistName* is the name of the existing subscription set that is being split, *Val* is either F or S, and *NewName* is the name of the new subscription set that you are creating.

4. From the Capture control server, run the following SQL statement to insert a new row for the new subscription set into the prune set (IBMSNAP_PRUNE_SET) table:

```
INSERT INTO Schema.IBMSNAP_PRUNE_SET
              (APPLY_QUALIFIER,
               SET_NAME,
               TARGET_SERVER,
               SYNCHTIME,
               SYNCHPOINT
       VALUES ('ApplyQual',
               'NewName',
               'Target_Server',
                NULL,
                x'00000000000000000000');
```

where *Schema* is the name of the Capture schema, *ApplyQual* is the Apply qualifier, *NewName* is the name of the new subscription set that you are creating, and *Target_Server* is the database location of the target tables.

5. From the Capture control server, run the following SQL statement to copy information from the existing subscription set row to the new subscription set row in the IBMSNAP_PRUNE_SET table:

```
UPDATE Schema.IBMSNAP_PRUNE_SET
   SET SYNCHPOINT        =
              (SELECT SYNCHPOINT FROM Schema.IBMSNAP_PRUNE_SET B
                WHERE APPLY_QUAL    = 'ApplyQual' AND
```

```
                    SET_NAME      = 'ExistName' AND
                    TARGET_SERVER = 'Target_Server'),
      SYNCHTIME =
            (SELECT SYNCHTIME FROM  Schema.IBMSNAP_PRUNE_SET B
              WHERE APPLY_QUAL    = 'ApplyQual' AND
                    SET_NAME      = 'ExistName' AND
                    TARGET_SERVER = 'Target_Server')
 WHERE
      APPLY_QUAL    = 'ApplyQual' AND
      SET_NAME      = 'NewName'   AND
      TARGET_SERVER = 'Target_Server';
```

where *Schema* is the name of the Capture schema, *ApplyQual* is the Apply qualifier, *ExistName* is the name of the existing subscription set that is being split, *Target_Server* is the database location of the target tables, and *NewName* is the name of the new subscription set that you are creating.

6. From the Apply control server, run the following SQL statements to change the subscription set name in the subscription members (IBMSNAP_SUBS_MEMBR) and the subscription columns (IBMSNAP_SUBS_COLS) tables for *each* subscription-set member that you are moving into the new subscription set:

```
UPDATE ASN.IBMSNAP_SUBS_MEMBR
   SET SET_NAME          = 'NewName'
 WHERE
      APPLY_QUAL        = 'ApplyQual' AND
      SET_NAME          = 'ExistName' AND
      WHOS_ON_FIRST     = 'Val'       AND
      SOURCE_OWNER      = 'SrcSchema' AND
      SOURCE_TABLE      = 'SrcTbl'    AND
      SOURCE_VIEW_QUAL  =  SrcVwQual  AND
      TARGET_OWNER      = 'TgtSchema' AND
      TARGET_TABLE      = 'TgtTbl';
UPDATE ASN.IBMSNAP_SUBS_COLS
   SET SET_NAME          = 'NewName'
 WHERE
      APPLY_QUAL        = 'ApplyQual' AND
      SET_NAME          = 'ExistName' AND
      WHOS_ON_FIRST     = 'Val'       AND
      TARGET_OWNER      = 'TgtSchema' AND
      TARGET_TABLE      = 'TgtTbl';
```

where *NewName* is the name of the new subscription set that you are creating, *ApplyQual* is the Apply qualifier, *ExistName* is the name of the existing subscription set being split, *Val* is either F or S, *SrcSchema* is the source table schema, *SrcTbl* is the source table name, *SrcVwQual* is the source-view qualifier for this source table, *TgtSchema* is the schema of the target table, and *TgtTbl* is the target table name.

Repeat this step for each subscription-set member that you want to move to the new subscription set.

7. If the subscription set that you are splitting uses before or after SQL statements or procedure calls, move the applicable statements to the new subscription set in the subscription statements (IBMSNAP_SUBS_STMTS) table:

   a. Run the following SQL script from the Apply control server to move the statements:

```
UPDATE ASN.IBMSNAP_SUBS_STMTS
   SET SET_NAME      = 'NewName'
 WHERE
       APPLY_QUAL    = 'ApplyQual' AND
```

```
SET_NAME      = 'ExistName' AND
WHOS_ON_FIRST = 'Val'       AND
STMT_NUMBER   in (Stmt1,Stmt2,..Stmtn);
```

where *NewName* is the name of the new subscription set that you are creating, *ApplyQual* is the Apply qualifier, *ExistName* is the name of the existing subscription set being split, *Val* is either F or S, and *Stmt1*, *Stmt2*, and *Stmtn* correspond to the numbers of the statements that you are moving to the new subscription set.

b. Adjust the AUX_STMTS column values in the IBMSNAP_SUBS_SET table to reflect the new count of statements for both subscription sets. Renumber the statements to eliminate any duplicates, if necessary.

8. From the Capture control server, run the following SQL statement to change the name of the subscription set in the pruning control (IBMSNAP_PRUNCNTL) table for *each* subscription-set member that you moved:

```
UPDATE Schema.IBMSNAP_PRUNCNTL
   SET SET_NAME          = 'NewName'
 WHERE
       APPLY_QUAL        = 'ApplyQual'      AND
       SET_NAME          = 'ExistName'      AND
       TARGET_SERVER     = 'Target_Server'  AND
       SOURCE_OWNER      = 'SrcSchema'      AND
       SOURCE_TABLE      = 'SrcTbl'         AND
       SOURCE_VIEW_QUAL  =  SrcVwQual       AND
       TARGET_OWNER      = 'TgtSchema'      AND
       TARGET_TABLE      = 'TgtTbl';
```

where *Schema* is the name of the Capture schema, *NewName* is the name of the new subscription set that you created in step 2, *ApplyQual* is the Apply qualifier, *ExistName* is the name of the existing subscription set that was split, *Target_Server* is the database location of the target tables, *SrcSchema* is the source table schema, *SrcTbl* is the source table name, *SrcVwQual* is the source-view qualifier for this replication source table, *TgtSchema* is the target table schema, and *TgtTbl* is the target table name.

Repeat this step for each subscription-set member that you moved to the new subscription set.

9. **For UNIX, Windows, z/OS:** If you are running the Apply program with **opt4one** set to y, stop and then restart the Apply program.
10. Reactivate both subscription sets using the following method:

Replication Center

From the Subscription Sets folder, right-click both deactivated subscription sets in the contents pane and select Activate. See the Replication Center help for details.

Merging subscription sets

If you want to merge two subscription sets into one, you can use the following procedure. You might want to merge subscription sets if you want the target tables within these two subscription sets to have the same transaction consistency but you do not want to delete and then re-create subscription set information.

Prerequisites:

Before running these SQL statements, familiarize yourself with the structure of the DB2 replication control tables and with the subscription sets defined on your system.

Identify the Capture control server, target server, and Apply control server of each subscription set that you want to merge. Verify that all of the subscription sets that you want to merge were created with the same Capture control server, target server, and Apply control server.

For UNIX, Windows, z/OS: If you set up monitoring definitions or started Replication Alert Monitor programs to detect alert conditions for the subscription sets that you are going to merge, drop these monitoring definitions. After you merge the subscription sets, re-create the monitoring definitions through the Replication Center. Then, you can reinitialize the associated Replication Alert Monitor programs using the **asnmcmd** system command with the **reinit** parameter. Alternatively, you can stop the Replication Alert Monitor programs using the **asnmcmd** system command with the **stop** parameter and then restart the programs using the **asnmon** system command.

Restriction:

The two subscription sets that you want to merge *must* derive their source data from the same Capture server and through the same Capture schema.

Procedure:

1. Use one of the following methods to stop the associated Capture program:

   Replication Center
   Use the Stop Capture window. See the Replication Center help for details.

   asnccmd system command (Windows, UNIX, z/OS)
   Use the **stop** parameter. See "asnccmd: Operating Capture" on page 288 for parameter descriptions and command syntax.

   ENDDPRCAP system command (OS/400)
   See "ENDDPRCAP: Stopping Capture (OS/400)" on page 364 for parameter descriptions and command syntax.

   Wait until both subscription sets reach the same synchpoint and synchtime as indicated in the subscription sets (IBMSNAP_SUBS_SET) table.

   Important: The two subscription sets *must* have processed the source data up to the identical synchpoint value to prevent a loss of data when the subscription sets are merged.

   Tip: If you do not want to stop the Capture program, insert a USER signal in the signal (IBMSNAP_SIGNAL) table, and generate an event with the END_SYNCHPOINT (in the subscriptions events [IBMSNAP_SUBS_EVENT] table) set to the value of the SIGNAL_LSN column in the IBMSNAP_SIGNAL table so that only the data up to that end point is applied.
2. Use the following method to deactivate the two subscription sets:

   Replication Center
   From the Subscription Sets folder, right-click the two active subscription sets in the contents pane and select Deactivate. See the Replication Center help for details.

3. From the Apply control server, run the following SQL statement to delete the row from the IBMSNAP_SUBS_SET table that corresponds to the subscription set that you are moving into the other subscription set:

```
DELETE FROM ASN.IBMSNAP_SUBS_SET
 WHERE
       APPLY_QUAL    = 'ApplyQual'      AND
       SET_NAME      = 'Subset_To_Move' AND
       WHOS_ON_FIRST = 'Val';
```

   where *ApplyQual* is the Apply qualifier, *Subset_To_Move* is the name of the subscription set that you are moving into another existing subscription set, and *Val* is either F or S.

4. From the Capture control server, run the following SQL statement to delete the row from the prune set (IBMSNAP_PRUNE_SET) table that corresponds to the subscription set that you are moving into the other subscription set:

```
DELETE FROM Schema.IBMSNAP_PRUNE_SET
 WHERE
       APPLY_QUAL    = 'ApplyQual'      AND
       SET_NAME      = 'Subset_To_Move' AND
       TARGET_SERVER = 'Target_Server' ;
```

   where *Schema* is the name of the Capture schema, *ApplyQual* is the Apply qualifier, *Subset_To_Move* is the name of the subscription set that you are moving into another existing subscription set, and *Target_Server* is the database location of the target tables.

5. From the Apply control server, run the following SQL statements to change the name of the subscription set that you are moving to the name of the other subscription set in the subscription members (IBMSNAP_SUBS_MEMBR) and subscription columns (IBMSNAP_SUBS_COLS) tables:

```
UPDATE ASN.IBMSNAP_SUBS_MEMBR
   SET SET_NAME      = 'Existing_Merged_Subset'
 WHERE
       APPLY_QUAL    = 'ApplyQual'      AND
       SET_NAME      = 'Subset_To_Move' AND
       WHOS_ON_FIRST = 'Val';

UPDATE ASN.IBMSNAP_SUBS_COLS
   SET SET_NAME      = 'Existing_Merged_Subset'
 WHERE
       APPLY_QUAL    = 'ApplyQual'      AND
       SET_NAME      = 'Subset_To_Move' AND
       WHOS_ON_FIRST = 'Val';
```

   where *Existing_Merged_Subset* is the name of the existing subscription set being merged with the subscription set that you are moving, *ApplyQual* is the Apply qualifier, *Subset_To_Move* is the name of the subscription set that you are moving into the existing subscription set, and *Val* is either F or S.

6. If the subscription set that you are moving uses before or after SQL statements or procedure calls, change the name of the subscription set in the subscription statements (IBMSNAP_SUBS_STMTS) table:

   a. Run the following SQL script from the Apply control server to change the name of the subscription set:

```
UPDATE ASN.IBMSNAP_SUBS_STMTS
   SET SET_NAME      = 'Existing_Merged_Subset'
 WHERE
       APPLY_QUAL    = 'ApplyQual'      AND
       SET_NAME      = 'Subset_To_Move' AND
       WHOS_ON_FIRST = 'Val';
```

where *Existing_Merged_Subset* is the name of the existing subscription set that is being merged with the subscription set that you are moving, *ApplyQual* is the Apply qualifier, *Subset_To_Move* is the name of the subscription set that you are moving into the existing subscription set, and *Val* is either F or S.

b. Adjust the AUX_STMTS column value in the IBMSNAP_SUBS_SET table to reflect the new count of statements in the existing merged subscription set. Renumber the statements to eliminate any duplicates, if necessary.

7. From the Capture control server, run the following SQL statement to change the name of the subscription set that was moved to the name of the merged subscription set in the pruning control (IBMSNAP_PRUNCNTL) table:

```
UPDATE Schema.IBMSNAP_PRUNCNTL
   SET SET_NAME      = 'Existing_Merged_Subset'
 WHERE
       APPLY_QUAL    = 'ApplyQual'      AND
       SET_NAME      = 'Subset_To_Move' AND
       TARGET_SERVER = 'Target_Server' ;
```

where *Schema* is the name of the Capture schema, *Existing_Merged_Subset* is the name of the existing subscription set being merged with the subscription set that you are moving, *ApplyQual* is the Apply qualifier, *Subset_To_Move* is the name of the subscription set that you are moving into another existing subscription set, and *Target_Server* is the database location of the target tables.

8. **For UNIX, Windows, z/OS:** If you are running the Apply program with **opt4one** set to y, stop and then restart the Apply program.

9. Reactivate the merged subscription set using the following method:

Replication Center

From the Subscription Sets folder, right-click the deactivated subscription set in the contents pane and select Activate. See the Replication Center help for details.

Changing Apply qualifiers of subscription sets

If you need to change the Apply qualifier of a subscription set, you can use SQL to make the change without deleting and re-creating the subscription set.

If you have several subscription sets using the same Apply qualifier, you might want to move some of the subscription sets to a new Apply qualifier to balance the work loads of the Apply programs.

You must run the SQL statements in this procedure for *each* subscription set that you want to move.

Prerequisites:

Before running these SQL statements, familiarize yourself with the structure of the DB2 replication control tables and with the subscription sets defined on your system.

You must also determine the following information:

- The name of the new Apply qualifier. (See Chapter 17, “Naming rules for SQL replication objects,” on page 269 for more information.)
- The subscription sets that you want to move from the existing Apply qualifier to the new Apply qualifier.

- Any before or after SQL statements or procedure calls that are defined for these subscription sets.

For UNIX, Windows, z/OS: If you set up monitoring definitions or started Replication Alert Monitor programs under the Apply qualifier that you are going to change, drop these monitoring definitions. After you change the Apply qualifier, re-create the monitoring definitions with the new Apply qualifier name through the Replication Center. Then, you can reinitialize the associated Replication Alert Monitor programs using the **asnmcmd** system command with the **reinit** parameter. Alternatively, you can stop the Replication Alert Monitor programs using the **asnmcmd** system command with the **stop** parameter and then restart the programs using the **asnmon** system command.

Procedure:

1. Deactivate the subscription sets that you want to change using the following method:

   Replication Center
   From the Subscription Sets folder, right-click the active subscription sets in the contents pane and select Deactivate. See the Replication Center help for details.

2. From the Apply control server, run the following SQL statements to change the Apply qualifier of the subscription set in the subscription sets (IBMSNAP_SUBS_SET), subscription members (IBMSNAP_SUBS_MEMBR), and subscription columns (IBMSNAP_SUBS_COLS) tables:

```
UPDATE ASN.IBMSNAP_SUBS_SET
   SET APPLY_QUAL    = 'NewApplyQual'
 WHERE
       APPLY_QUAL    = 'ExistApplyQual' AND
       SET_NAME      = 'Name'           AND
       WHOS_ON_FIRST = 'Val';
UPDATE ASN.IBMSNAP_SUBS_MEMBR
   SET APPLY_QUAL    = 'NewApplyQual'
 WHERE
       APPLY_QUAL    = 'ExistApplyQual' AND
       SET_NAME      = 'Name'           AND
       WHOS_ON_FIRST = 'Val';
UPDATE ASN.IBMSNAP_SUBS_COLS
   SET APPLY_QUAL    = 'NewApplyQual'
 WHERE
       APPLY_QUAL    = 'ExistApplyQual' AND
       SET_NAME      = 'Name'           AND
       WHOS_ON_FIRST = 'Val';
```

   where *NewApplyQual* is the new Apply qualifier, *ExistApplyQual* is the existing Apply qualifier, *Name* is the name of the subscription set, and *Val* is either F or S.

3. If this subscription set uses before or after SQL statements or procedure calls, run the following SQL script from the Apply control server to change the Apply qualifier of the subscription set in the subscription statements (IBMSNAP_SUBS_STMTS) table:

```
UPDATE ASN.IBMSNAP_SUBS_STMTS
   SET APPLY_QUAL    = 'NewApplyQual'
 WHERE
       APPLY_QUAL    = 'ExistApplyQual' AND
       SET_NAME      = 'Name'           AND
       WHOS_ON_FIRST = 'Val';
```

where *NewApplyQual* is the new Apply qualifier, *ExistApplyQual* is the existing Apply qualifier, *Name* is the name of the subscription set, and *Val* is either F or S.

4. From the Capture control server, run the following SQL statements to change the Apply qualifier of the subscription set in the prune set (IBMSNAP_PRUNE_SET) and pruning control (IBMSNAP_PRUNCNTL) tables:

```
UPDATE Schema.IBMSNAP_PRUNE_SET
   SET APPLY_QUAL    = 'NewApplyQual'
 WHERE
       APPLY_QUAL    = 'ExistApplyQual' AND
       SET_NAME      = 'Name'           AND
       TARGET_SERVER = 'Target_Server';

UPDATE Schema.IBMSNAP_PRUNCNTL
   SET APPLY_QUAL    = 'NewApplyQual'
 WHERE
       APPLY_QUAL    = 'ExistApplyQual' AND
       SET_NAME      = 'Name'           AND
       TARGET_SERVER = 'Target_Server';
```

   where *Schema* is the name of the Capture schema, *NewApplyQual* is the new Apply qualifier, *ExistApplyQual* is the existing Apply qualifier, *Name* is the name of the subscription set, and *Target_Server* is the database location of the target tables.

5. Repeat steps 2 through 4 for each remaining subscription set that you want to move.
6. **For UNIX, Windows, z/OS:** If you are running the Apply program with **opt4one** set to y, stop and then restart the Apply program.
7. Use the following method to reactivate the subscription sets:

   Replication Center
   From the Subscription Sets folder, right-click the deactivated subscription sets in the contents pane and select Activate. See the Replication Center help for details.

Deactivating subscription sets

You can deactivate a subscription set without removing it. When you deactivate a subscription set, the Apply program completes its current processing cycle and then suspends operations for that subscription set. You might need to perform special maintenance on these deactivated subscription sets depending on how long they must remain deactivated:

Short time-period
There are no special processing requirements for subscription sets that you temporarily deactivate. You should temporarily deactivate a subscription set while changing its attributes or while fixing failures on target tables.

Use the Replication Center to deactivate, change, and then reactivate a subscription set.

Longer time-period
You can deactivate a subscription set that you do not currently need but might want to use in the future. However, you must take additional action if this subscription set needs to remain deactivated for a time period that is long enough for changed data to accumulate and to affect the performance of the Capture and Apply programs.

The Capture program uses information from active Apply programs during the pruning process. If the Apply programs are inactive or the

subscriptions sets are deactivated for long periods of time, the pruning information becomes stale and the unit-of-work (UOW) and possibly the change-data (CD) tables cannot be pruned quickly and efficiently if active registrations that are associated with the deactivated subscription sets remain. This stale information can seriously degrade the performance of the remaining active Apply programs and cause unnecessary and costly CPU consumption by the pruning process. The UOW and CD tables are eventually pruned based on the retention limit (with a default value of seven days) of the Capture program. However, large amounts of data might accumulate during this time depending on the size of your replication environment.

To prevent these pruning problems, you can use SQL to reset the pruning information for a subscription set that must remain deactivated for a longer time-period.

Prerequisite:

Before running these SQL statements, familiarize yourself with the structure of the DB2 replication control tables and with the subscription sets defined on your system.

Procedure:

1. From the Replication Center, verify that the subscription set is not active.
2. From the Capture control server, run the following SQL statements to reset the pruning information in the prune set (IBMSNAP_PRUNE_SET) and pruning control (IBMSNAP_PRUNCNTL) tables for the deactivated subscription set:

```
UPDATE Schema.IBMSNAP_PRUNE_SET
   SET SYNCHPOINT    = x'00000000000000000000' AND
       SYNCHTIME     = NULL
 WHERE
       APPLY_QUAL    = 'ApplyQual' AND
       SET_NAME      = 'Name'      AND
       TARGET_SERVER = 'Target_Server';

UPDATE Schema.IBMSNAP_PRUNCNTL
   SET SYNCHPOINT    = NULL  AND
       SYNCHTIME     = NULL
 WHERE
       APPLY_QUAL    = 'ApplyQual' AND
       SET_NAME      = 'Name'      AND
       TARGET_SERVER = 'Target_Server';
```

   where *Schema* is the name of the Capture schema, *ApplyQual* is the Apply qualifier, *Name* is the name of the subscription set, and *Target_Server* is the database location of the target tables.

If you deactivated all the subscription sets associated with a registered object, you should also deactivate the registered object to prevent the Capture program from capturing data unnecessarily.

Removing subscription sets

If you no longer need to replicate the data in a particular subscription set, you can remove the subscription set. However, if your Apply program is processing the subscription set that you remove, your Apply program job abends and any other subscription sets in that job are not processed until you restart the job.

Procedure:

1. To ensure that the Apply program has completed any current processing for the subscription set, deactivate the subscription set before you remove it by using the following method:

   Replication Center
   From the Subscription Sets folder, right-click the active subscription set in the contents pane and select Deactivate. See the Replication Center help for details.

2. Use one of the following methods to remove a deactivated subscription set:

   Replication Center
   Use the Delete Subscription Set window. See the Replication Center help for details.

   RMVDPRSUB system command (OS/400)
   See "RMVDPRSUB: Removing a DPR subscription set (OS/400)" on page 381 for parameter descriptions and command syntax.

Important: The Capture program continues capturing data and writing rows to the change-data (CD) table even if you remove all subscription sets for the registered object. To prevent this continued processing by the Capture program, deactivate or remove the registered object after removing its subscription sets.

Coordinating replication events with database application events

You can coordinate database and replication events by manually inserting rows into the signal (IBMSNAP_SIGNAL) table. Manually inserted IBMSNAP_SIGNAL rows, known as signals, instruct running Capture programs to take specific actions.

Setting an event END_SYNCHPOINT using the USER type signal

You can set the SIGNAL_TYPE column value to USER to establish a precise point on the DB2 recovery log and to coordinate a replication event with a database application event.

For example, if you are replicating online transaction processing (OLTP) data to a separately maintained data warehouse, you might want to keep the warehouse data fairly stable for ad hoc query processing. So you update the warehouse data with only the changes that occurred up to a specific point in time in the OLTP application business day. In this case, the database application event is the logical end of the business day. The replication event would be the application of the changes from the close of business on one specific day to the close of business on the following day. Assume that the subscription sets are configured for event processing only.

Procedure:

To create a USER type signal:

1. Create a Capture USER type signal by inserting the following row into the IBMSNAP_SIGNAL table:

```
INSERT INTO Schema.IBMSNAP_SIGNAL
            (signal_type,
             signal_subtype,
```

```
        signal_state)
    VALUES('USER',
        'USER APPLY EVENT SIGNAL',
        'P');
```

Run this SQL INSERT statement when the database application event occurs (in this case at the end of the application business day).

The Capture program acts on this signal table log record after the Capture program finds this record on the database recovery log and only when the Capture program finds the corresponding commit record for this insert, verifying that this event was committed.

When a USER type signal is committed, the Capture program updates the following IBMSNAP_SIGNAL column values that correspond to the insert log record being processed:

- SIGNAL_STATE = 'R' (received by the Capture program)
- SIGNAL_LSN = the log sequence number from the commit log record for the DB2 unit of work that contains this signal row insert

2. Use the value that is now in the SIGNAL_LSN column of the inserted signal row an END_SYNCHPOINT value in the subscription events (IBMSNAP_SUBS_EVENT) control table. This new value alerts the Apply program that all the data for the new business day has been collected by the Capture program and that the Apply program should fetch and apply data only up to the value of the SIGNAL_LSN column.

You can automate the insert into the IBMSNAP_SUBS_EVENT table by creating an update trigger on the IBMSNAP_SIGNAL table:

```
CREATE TRIGGER EVENT_TRIG
    NO CASCADE AFTER UPDATE ON Schema.IBMSNAP_SIGNAL
       REFERENCING NEW AS N
          FOR EACH ROW MODE DB2SQL
              WHEN (N.SIGNAL_SUBTYPE = 'USER APPLY EVENT SIGNAL')
                    INSERT INTO ASN.IBMSNAP_SUBS_EVENT VALUES
                         ('WH_APPLY_EVENT',
                          (CURRENT TIMESTAMP + 2 MINUTES),
                           N.SIGNAL_LSN,
                           null);
```

This trigger fires each time that the IBMSNAP_SIGNAL table is updated by the Capture program. When a SIGNAL_SUBTYPE column is updated to 'USER APPLY EVENT SIGNAL', the trigger inserts a row into the IBMSNAP_SUBS_EVENT table. This row indicates to the Apply program that it must fetch and apply the work from the latest business day (which has been committed prior to the SIGNAL_LSN value as computed by the Capture program) after two minutes have elapsed.

Creating journal signal tables for remote journaling

On iSeries operating systems, a signal table is associated with each journal used for source tables. These tables are called journal signal tables and have the same structure as the global signal table schema.IBMSNAP_SIGNAL.

To use signals in a remote journaling environment, you need to create a journal signal table on the source system. You might also need to create a collection on the source system that has the same name as the capture schema that you are using on the Capture control server.

The name of journal signal table is schema.IBMSNAP_SIGNAL_*xxxx1_yyyy1*, where *xxxx1* is the journal library, and *yyyy1* is the journal name of the remote journal on the Capture control server. This table needs to be journaled to the source journal on the source server.

Procedure:

To journal a signal table to the source journal on the source server:

1. Ensure that a collection named after the Capture schema on the Capture control server exists.
2. Create the journal signal table schema.IBMSNAP_SIGNAL_xxxx1_yyyy1 as follows:

```
CREATE TABLE schema/IBMSNAP_SIGNAL_xxxx1_yyyy1
                            (SIGNAL_TIME TIMESTAMP NOT NULL WITH DEFAULT,
                             SIGNAL_TYPE VARCHAR(30) NOT NULL,
                             SIGNAL_SUBTYPE VARCHAR(30),
                             SIGNAL_INPUT_IN VARCHAR(500),
                             SIGNAL_STATE CHAR(1) NOT NULL,
                             SIGNAL_LSN CHAR(10) FOR BIT DATA)
```

3. End journaling using the End Journal Physical File (**ENDJRNPF**) command:

```
ENDJRNPF  FILE(schema/IBMSNnnnnn)
```

   where IBMSN*nnnnn* is the short name of IBMSNAP_SIGNAL_*xxxx1_yyyy1*
4. Start journaling the source journal using the Start Journal Physical File (**STRJRNPF**) command:

```
STRJRNPF  FILE(schema/IBMSNnnnnn)  JRN(xxxx2/yyyy2)
```

   where *xxxx2* is the journal library and *yyyy2* is the journal name of the source journal on the source server.

Using the Capture CMD STOP signal

You can set the SIGNAL_TYPE column value to CMD and the SIGNAL_SUBTYPE column value to STOP to stop a Capture program process at a precise point on the DB2 recovery log. There are two main uses of this capability:

- To coordinate the Capture program with any source table changes that render previous log records unreadable. This could occur if you dropped and then re-created a table or if you reorganized a table without setting the KEEPDICTIONARY option to YES.
- To coordinate a common recovery point between replicated distributed database systems.

Coordinating a source table change with the Capture program

You can use a Capture CMD type STOP subtype signal to shut down a Capture program and to coordinate source table changes.

Procedure:

To coordinate the source table changes:

1. Create a Capture CMD type STOP subtype signal by inserting a row into the signal (IBMSNAP_SIGNAL) table using the following SQL statement:

```
INSERT INTO Schema.IBMSNAP_SIGNAL
            (signal_type,
             signal_subtype,
             signal_state)
      VALUES('CMD',
             'STOP',
             'P');
```

You should insert this row when the database application event occurs, after the source table activity has been quiesced but prior to the activity that causes problematic log record changes.

The Capture program acts on this signal table log record after the Capture program finds this record on the database recovery log and only when the Capture program finds the corresponding commit record for this insert, verifying that this event was committed.

The Capture program shuts down all Capture threads in an orderly manner after committing all captured data from the transactions on the log that are prior to the commit log record for the DB2 unit of work that contains this inserted IBMSNAP_SIGNAL row. Before terminating, the Capture program also updates the following values in the IBMSNAP_SIGNAL table row that corresponds to the insert log record being processed:

- SIGNAL_STATE = 'R' (received by the Capture program)
- SIGNAL_LSN = the log sequence number from the commit log record for the DB2 unit of work that contains this signal row insert

All log records for the changing source table are processed by the Capture program when it terminates.

2. Depending on your scenario, drop and re-create your source table, or reorganize and compress your source table without set the KEEPDICTIONARY option to YES.
3. If you dropped or altered replicated columns, you should now alter the corresponding registrations and subscription sets that you created for this source table. Such changes, if necessary, can be further coordinated with the Apply program by waiting for the affected subscription sets to catch up to the currently stopped Capture program. A subscription set is in synch with the Capture program when the SYNCHPOINT column value in the subscription sets (IBMSNAP_SUBS_SET) table is equal to the MAX_COMMITSEQ column value in the *Schema*.IBMSNAP_RESTART table.

Setting a distributed recovery point

You can use a Capture CMD type STOP subtype signal to set your source and target databases to equivalent recovery points and recover the databases at a common point of consistency.

Prerequisites:

Before using this procedure, verify that your Apply control tables have been created in the target database.

Also, verify that all activity against the source database has been quiesced before inserting the row into the IBMSNAP_SIGNAL table. However, do not create the backup or image copy of the database tables until after you insert the row into the IBMSNAP_SIGNAL table.

If your subscription sets are not typically configured for event processing, then you must temporarily set your subscription sets to event-based timing. Use the following SQL statement to insert a row into the subscription events (IBMSNAP_SUBS_EVENT) table:

```
INSERT INTO ASN.IBMSNAP_SUBS_EVENT
       VALUES('RECOVERY_EVENT',
              CURRENT TIMESTAMP + 2 MINUTES,
              SIGNAL_LSN_value,
              NULL);
```

where *SIGNAL_LSN_value* is the log sequence number set by the Capture program and stored in the IBMSNAP_SIGNAL table.

Procedure:

To set a distributed recovery point:

1. Create a Capture CMD type STOP subtype signal by inserting a row into the IBMSNAP_SIGNAL table using the following SQL statement:

   ```
   INSERT INTO Schema.IBMSNAP_SIGNAL
              (signal_type,
               signal_subtype,
               signal_state)
         VALUES('CMD',
                'STOP',
                'P');
   ```

   The Capture program acts on this signal table log record after the Capture program finds this record on the database recovery log and only when the Capture program finds the corresponding commit record for this insert, verifying that this event was committed.

   The Capture program shuts down all Capture threads in an orderly manner after committing all captured data from the transactions on the log that is prior to the commit log record for the DB2 unit of work that contains this inserted IBMSNAP_SIGNAL row. Before terminating, the Capture program also updates the following values in the IBMSNAP_SIGNAL table row that corresponds to the insert log record being processed:

   - SIGNAL_STATE = 'R' (received by the Capture program)
   - SIGNAL_LSN = the log sequence number from the commit log record for the DB2 unit of work that contains this signal row insert

   All log records for the source database are processed by the Capture program when it terminates.
2. Run the source database backup or image copy utilities.
3. Use the value in the SIGNAL_LSN column from the IBMSNAP_SIGNAL table row that you inserted as an END_SYNCHPOINT value in the IBMSNAP_SUBS_EVENT table. This value alerts the Apply program that all the data committed prior to the backup point has been collected by the Capture program and that the Apply program should fetch and apply data only up to the value of the SIGNAL_LSN column.

   The subscription sets process all data up to the SIGNAL_LSN value.
4. Run the target database backup or image copy utilities. The source and target databases now have equivalent recovery points, and you can recover both databases at a common point of consistency.

You can resume all source database activity as soon as the Apply events have been set and the source database backup or image copy utility activity is complete. You can also start the Capture program. After the target database backup or image copy utility activity is complete, you can change the scheduling options of your subscription sets back to their original settings (time-based, event-based, or both).

On iSeries operating systems, you can send the STOP signal to stop a single journal job or to stop all the journal jobs. To stop a single journal job, insert the signal into the signal table designated for that journal (the IBMSNAP_SIGNAL_*xxxx_yyyy* table, where *xxxx* is the journal library and *yyyy* is the journal name. To stop all the journal jobs, insert the signal into table *schema*.IBMSNAP_SIGNAL. To stop a single journal job in a remote journal

configuration, insert the signal into the journal signal table on the source server. See “Creating journal signal tables for remote journaling” on page 197 for a description of how to create journal signal tables in a remote journal configuration.

Performing a CAPSTART handshake signal outside of the Apply program

Before any subscription set can be used by the Apply program to fetch and apply changes from the CD tables, there must be a ″handshake″ (synchronized communication) between the Capture and Apply programs of each subscription-set member in that subscription set.

The Apply program initiates the handshake by inserting a CMD type CAPSTART subtype signal into the signal (IBMSNAP_SIGNAL) table. The Apply program inserts this signal before performing a full refresh of any subscription-set member with a target table that is defined as complete.

Procedure:

To perform a CAPSTART handshake signal:

- Create a Capture CMD type CAPSTART subtype signal by inserting a row into the IBMSNAP_SIGNAL table using the following SQL statement:

```
INSERT INTO Schema.IBMSNAP_SIGNAL
            (signal_type,
             signal_subtype,
             signal_input_in,
             signal_state)
       VALUES('CMD',
              'CAPSTART',
              mapid,
             'P');
```

  where *mapid* is the MAP_ID column value of the *Schema*.IBMSNAP_PRUNCNTL table and corresponds to the row for the subscription-set member requiring the handshake.

Note: Run this SQL INSERT statement before performing a full refresh of the subscription-set member, if necessary.

The Capture program acts on this signal table log record after the Capture program finds this record on the database recovery log and only when the Capture program finds the corresponding commit record for this insert, verifying that this event was committed.

The Capture program checks if it already placed the associated registration into memory based on prior use of the registered table. If the registered table is not in use, the Capture program reads the associated registration information into memory and sets values in the register (IBMSNAP_REGISTER) table to show that this registered table is now active and in use.

Regardless of whether or not the registered table is in use, the Capture program sets the values of the SYNCHPOINT and SYNCHTIME columns in the associated row in the *Schema*.IBMSNAP_PRUNCNTL table to the log sequence number from the commit log record for the DB2 unit of work that contains this inserted signal row and to the timestamp from that same commit log record, respectively.

The Capture program updates the following values in the IBMSNAP_SIGNAL table row that corresponds to the insert log record being processed:

- SIGNAL_STATE = 'C' (received and completed by the Capture program)
- SIGNAL_LSN = the log sequence number from the commit log record for the DB2 unit of work that contains this signal row insert

Performing a CAPSTOP signal

You can initiate a CAPSTOP signal if you want to manually stop capturing changes for a registration. You can use this signal when deactivating a registration or before you remove a registration.

Procedure:

To perform a CAPSTOP signal:

1. Create a Capture CMD type CAPSTOP subtype signal by inserting a row into the IBMSNAP_SIGNAL table using the following SQL statement:

```
INSERT INTO Schema.IBMSNAP_SIGNAL
            (signal_type,
             signal_subtype,
             signal_input_in,
             signal_state)
       VALUES('CMD',
              'CAPSTOP',
               source_owner.source_table,
              'P');
```

   where *Schema* is the name of the Capture schema and *source_owner.source_table* is the fully qualified name of the table that no longer requires captured changes.

   The Capture program acts on this signal table log record after the Capture program finds this record on the database recovery log and only when the Capture program finds the corresponding commit record for this insert, verifying that this event was committed.

   The Capture program checks if it has already placed the associated registration into memory based on prior use of the registered table. If the registered table is not currently in use, the Capture program ignores the CAPSTOP signal.

   If the registered table is in use, the Capture program clears the memory associated with this registration and inactivates the registration (by setting the STATE column in the IBMSNAP_REGISTER table to 'I'). The Capture program then stops capturing changes for this registered table.

   The Capture program updates the following column values in the IBMSNAP_SIGNAL table row that corresponds to the insert log record being processed:

   - SIGNAL_STATE = 'C' (received and completed by the Capture program)
   - SIGNAL_LSN = the log sequence number from the commit log record for the DB2 unit of work that contains this signal row insert

2. Optional: Remove the registration.

On iSeries operating systems, you can also send a CAPSTOP signal to stop capturing changes for a registration by inserting the signal into the IBMSNAP_SIGNAL_*xxxx_yyyy* table, where *xxxx* is the journal library and *yyyy* is the journal name of the subject journal. To stop capturing changes for a registration in a remote journal configuration, insert the CAPSTOP signal on the source server.

See “Creating journal signal tables for remote journaling” on page 197 for a description of how to create journal signal tables in a remote journal configuration.

Promoting your replication configuration to another system

When you define registered objects or subscription sets on one system (a test system, for example), and you need to copy the replication environment to another system (a production system, for example), you can use the promote functions of the Replication Center. These functions reverse engineer your registered objects or subscription sets to create script files with appropriate data definition language (DDL) and data manipulation language (DML). You can copy the replication definitions to another database without having to re-register the sources or re-create the subscription sets.

For example, use the promote functions to define subscription sets for remote target databases. After you define a model target system in your test environment, you can create subscription-set scripts (and modify which Apply qualifier is used and so on) for your remote target systems, which are not otherwise supported from a central control point.

Important: The promote functions do not connect to the destination target system and do not validate the replication configuration parameters of that system.

There are three promote functions:

Promote registered tables
This function promotes registration information for specified tables. This function optionally promotes base table, index and table space definitions. You can specify a different Capture schema and a different server name for the tables that you promote. Also, you can change the schema name for the change-data (CD) tables associated with the promoted source tables.

You can promote multiple registered tables at one time. The new schema names that you provide are applied to all the promoted tables.

This function promotes tables that are registered under DB2 Universal Database Version 8 only.

Promote registered views
This function promotes registration information for specified views. This function optionally promotes base view, unregistered base table (on which a view is based), index, and table space definitions. You can specify a different Capture schema and a different server name for the views that you promote. Also, you can change the schema name for the CD views that are associated with the promoted source views and the CD tables on which these CD views are based.

You can promote multiple registered views at one time. The new schema names that you provide are applied to all the promoted views.

Important: If the view that you are promoting is based on a registered source table, you must promote the registered source table separately by using the promote registered tables function. These registered source tables are not automatically promoted by the promote registered view function. However, the unregistered base tables, upon which this view is based, are promoted by this function, if required.

Promote subscription sets

This function promotes subscriptions sets. This function enables you to copy a subscription set (with all of its subscription-set members) from one database to another.

You should use the promote subscription sets function with the promote registered tables function.

Important: You can use the promote functions to promote registered objects and subscription sets that reside on OS/400, UNIX, Windows, and z/OS operating systems. The promote functions copy replication definitions between like systems only, for example from one DB2 Universal Database for z/OS system to another DB2 Universal Database for z/OS system.

You cannot use the promote functions to copy replication definitions to or from non-DB2 relational databases. Additionally, you cannot use the promote functions to copy replication definitions that include OS/400 remote journals.

Related concepts:

- Chapter 15, "Using the Replication Center for SQL replication," on page 219

Related tasks:

- Chapter 3, "Registering tables and views as SQL replication sources," on page 35
- Chapter 4, "Subscribing to sources for SQL replication," on page 57

Related reference:

- "*schema*.IBMSNAP_SIGNAL" on page 458

Chapter 14. Maintaining an SQL replication environment

This chapter explains how to maintain the source systems, control tables, and target tables that reside on your database and are used by DB2 replication.

DB2 replication works with your database system and requires limited changes to your existing database activities. However, to ensure that your entire system continues to run smoothly and to avoid potential problems, you should determine the processing requirements of your replication environment and the potential impact of these requirements on your database system. This chapter discusses the maintenance requirements of these three functional components of DB2 replication:

- "Maintaining your source systems"
- "Maintaining your control tables" on page 209
- "Maintaining your target tables" on page 216

Maintaining your source systems

The replication source system contains the change-capture mechanism, the source tables that you want to replicate (including any remote journals used on OS/400 systems), the log data used by the Capture program, and any Capture triggers used on non-DB2 relational database sources. This section explains how to maintain your source tables and log files properly and how to ensure that these tables and files are always accessible to DB2 replication.

Maintaining source objects

Replication source objects are database tables and views that require the same maintenance as other database tables and views on your system. Continue to run your existing utilities and maintenance routines on these objects.

You need to consider the availability of these source tables to DB2 replication so that the Capture and Apply programs are always able to proceed. DB2 replication does not require direct access to source tables during most replication processing. However, DB2 replication *must* access your source tables or table spaces directly when one of the following two actions occurs:

- The Apply program performs a full refresh.
- The log manager attempts to read compressed log records (z/OS only).

Make sure that read access is available to your source tables to avoid disrupting your replication Apply program processing during a full refresh. Also, on z/OS, make sure that your utilities run in an online mode so that DB2 can obtain a latch against the compressed log record table space if your source tables are compressed. If your utilities and maintenance routines run in an exclusive mode that requires your database (or the compressed table space on z/OS) to be taken offline, your source objects will be unavailable to replication.

Maintaining and retaining source logs and journal receivers

Your DB2 recovery logs serve two purposes: to provide DB2 recovery capabilities and to provide information to your running Capture programs. You need to retain log data for both DB2 recovery and for DB2 replication, and you must be

absolutely certain that the Capture programs and DB2 are completely finished with a set of logs or journal receivers before you delete this data.

Note: DB2 replication does not use log data from non-DB2 relational databases.

Retaining log data (Linux, UNIX, Windows, z/OS)

Log data resides in log buffers, active logs, or archive logs. Each time the Capture program warm starts it requires all the DB2 logs created since it stopped as well as any DB2 logs that it did not completely process.

For Linux, UNIX and Windows: You must configure your database to use user-exit archiving for your Capture programs to retrieve data from archived logs.

If you run the Capture program whenever DB2 is running, the Capture program is typically up to date with the recovery logs of DB2. If you run Capture programs whenever DB2 is up or you retain log records for a week or longer, you can continue to use your existing log retention procedures. However, you should change your log retention procedures to accommodate DB2 replication if:

- You typically delete log records as soon as DB2 completes a backup, and these log records are no longer needed for forward recovery.
- You face storage constraints and need to delete your archived recovery logs frequently.

Procedure:

To determine which log records must be retained for use by the Capture program and which log records can be deleted:

On Linux, UNIX and Windows:

1. Run the following SQL statement to obtain the MIN_INFLIGHTSEQ value from the restart (IBMSNAP_RESTART) table:

   ```
   SELECT MIN_INFLIGHTSEQ
   FROM ASN.IBMSNAP_RESTART
   WITH UR;
   ```

   The MIN_INFLIGHTSEQ value appears. (There is only one row in the IBMSNAP_RESTART table. In a multi-partitioned environment, this procedure must be extended to each partition because each partition maintains its own set of log files. Use the SEQUENCE column from the IBMSNAP_PARTITIONINFO table to determine this information for each partition.) The MIN_INFLIGHTSEQ value is a char(10) for bit data column, which looks like 20 hexadecimal characters. For example:

   00000000123456123456

   Make note of the *last* 12 characters of the MIN_INFLIGHTSEQ value. In the example:

   123456123456

2. From a command line, type the **db2 get db cfg** command to obtain the path for the active log files. For example:

   ```
   db2 get db cfg for yourdbname
   ```

   where *yourdbname* is the database name. From the output displayed on the screen, note the path for the active log files. For example:

```
Path to log files   =C:\DB2\NODE0000\SQL00001\SQLLOGDIR\
```

3. From a DB2 command line, type the **db2flsn** command and enter the *last* 12 characters of the MIN_INFLIGHTSEQ value. For example:

   ```
   C:\DB2\NODE0000\SQL00001\>db2flsn 123456123456
   ```

   To run the **db2flsn** command, you must have access to the SQLLOGCTL.LFH file, which is located one directory above (C:\DB2\NODE0000\SQL00001\) the path to active log files.

   The system retrieves and displays the name of the file that contains the log record identified by the log sequence number. For example:

   ```
   Given LSN is contained in the log file S000123.LOG
   ```

4. Note the age of this retrieved log file.

   The Capture program needs this log file and more recent log files to perform a restart at any particular time. You must retain this log file and more recent log files but can delete any older logs to ensure continuous operation of the Capture programs.

On z/OS:

1. Run the following SQL statement to obtain the MIN_INFLIGHTSEQ value from the restart (IBMSNAP_RESTART) table:

   ```
   SELECT MIN_INFLIGHTSEQ
   FROM ASN.IBMSNAP_RESTART
   WITH UR;
   ```

   The MIN_INFLIGHTSEQ value appears. (There is only one row in the IBMSNAP_RESTART table.) For example:

   0000555551F031230000

   Ignore the first four characters, which are always 0000. The next 12 characters correspond to the active log sequence number. (This 12–character value is the relative byte address [RBA] in non-data sharing environments and is the log record sequence number [LRSN] in data sharing environments.) The last four characters are 0000 in non-data sharing environments; these last four characters correspond to the member ID in data sharing environments.

2. Use the DSNJU004 utility to invoke the Print Log Map utility. This utility displays information about the bootstrap data sets (BSDS).

   For example:

   ```
   #  ACTIVE LOG COPY 1 DATA SETS
   #   START RBA/TIME    END RBA/TIME         DATE   LTIME   DATA SET INFORMATION
   #-----------------  --------------      -------- ------ ------------------------
   # 555551F03000         555551F05FFF      1998.321 12:48  DSN=DSNC710.LOGCOPY1.DS02
   #2001.57 15:46:32.2  2001.057 15:47:03.9 PASSWORD=(NULL)STATUS=TRUNCATED,REUSABLE
   # 555551F06000         555551F09FFF      1998.321 12:49  DSN=DSNC710.LOGCOPY1.DS03
   #2001.57 15:47:32.2  2001.057 15:48:12.9 PASSWORD=(NULL)STATUS=TRUNCATED,REUSABLE
   ```

3. Compare your 12–character active log number of the MIN_INFLIGHTSEQ value to the Start RBA and corresponding End RBA range in each displayed row.
4. Find the row in which the value of your 12–character active log number exists. In the example:

```
# 555551F03000         555551F05FFF      1998.321 12:48  DSN=DSNC710.LOGCOPY1.DS02
#2001.57 15:46:32.2  2001.057 15:47:03.9 PASSWORD=(NULL)STATUS=TRUNCATED,REUSABLE
```

5. Note the corresponding Data Set Information for that active log number. In the example:

   ```
   DSNC710.LOGCOPY1.DS02
   ```

6. Note the date and time of this data set.

   The Capture program needs this data set and more recent data sets to perform a restart at any particular time.

Consider the age of this log file or data set to be a benchmark. You must retain this file and more recent log files but can delete any older logs to ensure continuous operation of the Capture programs.

Recommendation: Run the Capture program whenever DB2 is up to gain optimal performance, because the Capture program reads the log records directly from the log buffers.

Retaining journal receivers (OS/400)

It is important to retain all journal receivers that are required by the Capture program. When you restart the Capture program with the RESTART(*YES) parameter, the Capture program continues processing from where it ended previously and requires all the journal receivers used by one or more of the source tables.

To make certain your Capture program can access all required journal receivers, use the delete journal receiver exit program, which was registered automatically when you installed DB2 DataPropagator for iSeries. This exit program is invoked any time you or one of your applications programs attempts to delete a journal receiver. This exit program then determines whether or not a journal receiver can be deleted.

Recommendation: Specify DLTRCV(*YES) and MNGRCV(*SYSTEM) on the **CHGJRN** or **CRTJRN** command to use the delete journal receiver exit program and leave journal management to the system.

If the journal receiver is used by one or more source tables, the delete journal receiver exit program checks that the receiver being deleted does not contain entries that have not been processed by the Capture program. The exit program *disapproves* the deletion of the receiver if the Capture program still needs to process entries on that receiver. For more information, see “Managing journals and journal receivers (OS/400)” on page 32.

Using compression dictionaries (z/OS)

If you are using DB2 compression dictionary utilities, you must coordinate the use of these utilities with your Capture programs.

Updating DB2 compression dictionaries (z/OS)

When the Capture program requests log records, DB2 must decompress the log records of any table stored in a compressed table space. DB2 uses the current compression dictionary for decompression. If the compression dictionary is temporarily unavailable, DB2 returns an error to the Capture program. The Capture program makes several attempts to continue

processing. But if the dictionary remains unavailable, the Capture program issues an ASN0011E message and terminates. Alternatively, if the compression dictionary is no longer available, the Capture program deactivates the registration. To avoid these situations, let the Capture program process all log records for a table before performing any activity that affects the compression dictionary for that table. These activities include:

- Altering a table space to change its compression setting
- Using DSN1COPY to copy compressed table spaces from one subsystem to another, including from data sharing to non-data-sharing environments
- Running the REORG utility on the table space

Recommendation: Use the KEEPDICTIONARY=YES option to retain the current version of the compression dictionary during a reorganization. The KEEPDICTIONARY=YES option ensures that your dictionary remains compatible with your pre-existing log records.

If, however, you prefer to generate a new compression dictionary, synchronize the REORG utility with your current running applications and with the Capture program as follows:

1. Quiesce all application programs that update the table.
2. Let the Capture program capture all logged updates for the table.
3. Use the REORG utility on the compressed table to create a new compression dictionary.
4. Restart your application programs.

Latching DB2 compression dictionaries (z/OS)

You should also consider the availability of your compression directory. When the Capture program reads compressed log records, DB2 takes a latch on the source compressed table space to access the dictionary. The Capture program stops if the compressed table space on the source system is in the STOPPED state when the DB2 Log Read Interface needs this latch. Conversely, a utility that requires complete access to the source table space or that requires the table space to be in a STOPPED state can be locked out by the latch held by the Capture program while it is reading the dictionary.

To prevent any temporary lockout due to an unavailable latch, suspend the Capture program when a source compressed table space needs to be used exclusively by a DB2 (or vendor) utility.

Maintaining your control tables

DB2 replication uses control tables to store source definitions, subscription-set definitions, and other replication-specific control information. Although the size of some control tables remains static, other control tables can grow (and later shrink) dynamically depending on the size of your database and your replication requirements.

The size of the following control tables changes frequently during normal processing:

- Apply job (IBMSNAP_APPLY_JOB) (OS/400 only)
- Apply trace (IBMSNAP_APPLYTRACE)
- Apply trail (IBMSNAP_APPLYTRAIL)
- Capture monitor (IBMSNAP_CAPMON)

- Capture trace (IBMSNAP_CAPTRACE)
- Change data (*schema.CD_table*)
- Consistent-change data (*schema.target_table*)
- Replication Alert Monitor alerts (IBMSNAP_ALERTS)
- Replication Alert Monitor trace (IBMSNAP_MONTRACE)
- Replication Alert Monitor trail (IBMSNAP_MONTRAIL)
- Signal (IBMSNAP_SIGNAL)
- Subscription events (IBMSNAP_SUBS_EVENT)
- Unit-of-work (IBMSNAP_UOW)

The size and growth of these dynamic control tables can affect the performance of your system.

This section discusses maintenance activities that you should perform on your control tables.

Using the RUNSTATS utility (Linux, UNIX, Windows, z/OS)

The RUNSTATS utility updates statistics about the physical characteristics of your tables and associated indexes. You should continue to run the RUNSTATS utility on your existing tables at the same frequency as before you used DB2 replication. However, you should run the RUNSTATS utility on your change-data (CD), unit-of-work (IBMSNAP_UOW), and other dynamic control tables only one time when these tables contain substantial amounts of data. RUNSTATS reports meaningful information about these dynamic tables when these tables are at their maximum production-level size, and the optimizer gains the necessary statistics to determine the best strategy for accessing data.

Rebinding packages and plans (Linux, UNIX, Windows, z/OS)

Many of the DB2 replication packages and plans are bound using isolation UR (uncommitted reads). If you must rebind your packages and plans, note that your internal maintenance programs used for automatic rebinding of these packages and plans can cause contention problems between Capture and Apply if these programs rebind the replication packages with standard options such as cursor stability. DB2 replication packages must remain bound with isolation UR to maintain optimal system performance.

See “Setting up the replication programs” on page 25 for more information.

Reorganizing your control tables

You should regularly reorganize dynamic control tables that are frequently updated. Your change-data (CD) and unit-of-work (IBMSNAP_UOW) tables receive many INSERTS during change capture and many DELETES during pruning. The size of the Capture monitor (IBMSNAP_CAPMON), Capture trace (IBMSNAP_CAPTRACE), and Apply trail (IBMSNAP_APPLYTRAIL) tables can change dramatically depending on the update rates of your replication source tables.

Procedure:

Use one of the following table reorganization methods to eliminate fragmented data and reclaim space:

REORG command (Linux, UNIX, Windows)

REORG utility with the PREFORMAT option (z/OS)
The PREFORMAT option of this utility speeds up the insert processing of the Capture program.

RGZPFM (Reorganize Physical File Member) command (OS/400)
You can reorganize the UOW table and active CD tables when the Capture program ends by specifying the **RGZCTLTBL(*YES)** parameter on the **ENDDPRCAP** command. (See "ENDDPRCAP: Stopping Capture (OS/400)" on page 364 for command syntax and parameter descriptions.)

Recommendation: Reorganize the following dynamic control tables once a week:

- CD tables
- IBMSNAP_ALERTS
- IBMSNAP_APPLYTRACE
- IBMSNAP_APPLYTRAIL
- IBMSNAP_CAPMON
- IBMSNAP_CAPTRACE
- IBMSNAP_MONTRAIL
- IBMSNAP_MONTRACE
- IBMSNAP_UOW

You do not need to run any utilities that reclaim unused space or generate frequently updated optimizer statistics on the static control tables:

- Apply enqueue (IBMSNAP_APPENQ)
- Apply parameters (IBMSNAP_APPPARMS)
- Capture enqueue (IBMSNAP_CAPENQ) (Linux, UNIX, Windows, z/OS)
- Capture parameters (IBMSNAP_CAPPARMS)
- Capture Partition information (IBMSNAP_PARTITIONINFO)
- Capture schemas (IBMSNAP_CAPSCHEMAS)
- Prune lock (IBMSNAP_PRUNE_LOCK)
- Prune set (IBMSNAP_PRUNE_SET)
- Pruning control (IBMSNAP_PRUNCNTL)
- Register (IBMSNAP_REGISTER)
- Register extension (IBMSNAP_REG_EXT) (OS/400 only)
- Register synchronization (IBMSNAP_REG_SYNCH)
- Replication Alert Monitor conditions (IBMSNAP_CONDITIONS)
- Replication Alert Monitor contacts (IBMSNAP_CONTACTS)
- Replication Alert Monitor contact groups (IBMSNAP_CONTACTGRP)
- Replication Alert Monitor enqueue (IBMSNAP_MONENQ)
- Replication Alert Monitor groups (IBMSNAP_GROUPS)
- Replication Alert Monitor parameters (IBMSNAP_MONPARMS)
- Replication Alert Monitor servers (IBMSNAP_MONSERVERS)
- Restart (IBMSNAP_RESTART)
- Sequencing (IBMSNAP_SEQTABLE)
- Subscription columns (IBMSNAP_SUBS_COLS)
- Subscription members (IBMSNAP_SUBS_MEMBR)
- Subscription sets (IBMSNAP_SUBS_SET)
- Subscription statements (IBMSNAP_SUBS_STMTS)

Pruning your control tables

You should regularly prune your replication control tables to remove obsolete data and to improve system performance. This section discusses the different methods that you can use to prune your control tables and how these methods affect the performance of your system.

Pruning dynamic control tables maintained by the Capture programs

You should monitor the growth of and consider the various pruning methods available for the following dynamic control tables:

- CD tables
- IBMSNAP_UOW
- IBMSNAP_CAPMON
- IBMSNAP_CAPTRACE
- IBMSNAP_SIGNAL
- IBMSNAP_AUTHTKN (OS/400 only)

You can set your Capture programs to prune these tables automatically at regular intervals. Or you can prune on demand by launching the pruning process once; the Capture program does not prune again until you enter the next prune command.

Recommendation: Consider the use of automatic pruning to manage the growth of these control tables. Automatic pruning minimizes storage costs, increases the efficiency of your Apply programs, and generally reduces the risk of system failure due to storage overflow by regularly removing obsolete data from these tables. To invoke automatic pruning:

- Set the Capture program **autoprune** parameter to y (Linux, UNIX, Windows, z/OS).
- Use the **CLNUPITV(*IMMED)** or **CLNUPITV(*DELAYED)** Capture program parameter setting (OS/400).

With autopruning, you set the **prune_interval** operational parameter (on Linux, UNIX, Windows, and z/OS) or the **RETAIN** parameter (on OS/400) to specify how frequently the automatic pruning process occurs.

Procedure:

Use one of the following methods to initiate pruning:

Replication Center
Use the Prune Capture Control Tables window to prune the tables once. See the Replication Center help for details.

asncap system command with autoprune=y (Linux, UNIX, Windows, z/OS)
Use this command to start a Capture program with automatic pruning. See "asncap: Starting Capture" on page 282 for command syntax and parameter descriptions.

asnccmd system command with chgparms autoprune=y (Linux, UNIX, Windows, z/OS)
Use this command to enable automatic pruning in a running Capture program. See "asnccmd: Operating Capture" on page 288 for command syntax and parameter descriptions.

asnccmd system command with the prune parameter (Linux, UNIX, Windows, z/OS) Use this command to initiate pruning once from a running Capture program. See "asnccmd: Operating Capture" on page 288 for command syntax and parameter descriptions.

STRDPRCAP CLNUPITV(*IMMED) or STRDPRCAP CLNUPITV(*DELAYED) system commands (OS/400)
Use these commands to prune old rows at specified intervals after starting a Capture program. See "STRDPRCAP: Starting Capture (OS/400)" on page 393 for parameter descriptions and command syntax.

OVRDPRCAPA PRUNE(*IMMED) or OVRDPRCAPA PRUNE(*DELAYED) system command (OS/400)
Use this command to change the way that a running Capture program prunes these control tables. See "OVRDPRCAPA: Overriding DPR capture attributes (OS/400)" on page 375 for command syntax and parameter descriptions.

Pruning the CD and UOW tables: During each pruning cycle, whether invoked automatically or on demand, the Capture program prunes the CD and UOW tables based on the progress reported by the Apply programs. Progress is indicated by the SYNCHPOINT column value in the prune set (IBMSNAP_PRUNE_SET) table. This *normal* pruning is based on the minimum synchpoint value over all Apply programs that subscribe to each CD table and on the minimum overall synchpoint value for the UOW table.

Normal pruning, however, does not prune the CD and UOW tables effectively if the associated subscriptions sets run very infrequently. Keep pruning effectiveness in mind when deciding how often to run the associated Apply programs, when stopping these Apply programs, and when deactivating the subscription sets for more than a brief period of time.

If you run your subscription sets very infrequently or stop your Apply programs, your CD and UOW tables can grow very large and become eligible for retention limit pruning. The retention limit is an operational parameter of the Capture program, with a default value of one week. It determines how long old data remains in the tables before becoming eligible for retention limit pruning.

If the normal pruning process is inhibited due to deactivated or infrequently run subscription sets, data can remain in the table for long periods of time. If this data becomes older than the current DB2 timestamp minus the retention limit value, the retention limit pruning process prunes this data from the tables.

Try to avoid conditions that require retention limit pruning, because the accumulation of old data can lead to storage overflows and performance degradation. See "Deactivating subscription sets" on page 194 for more information.

Recommendation: Run your Apply programs at least once per day for all of your subscription sets.

If the source server is supplying changed data to a variety of target systems, each with very different requirements and some with infrequently running Apply programs for few registered sources, consider the use of multiple Capture programs. You can use multiple Capture programs and manage the various processing requirements with different Capture schemas, using one Capture

schema to isolate those tables that are infrequently pruned due to specific subscription-set timing requirements and using another Capture schema for the remaining source tables.

Pruning the Capture monitor and Capture trace tables: During each pruning cycle, the Capture program prunes the Capture monitor (IBMSNAP_CAPMON) and the Capture trace (IBMSNAP_CAPTRACE) tables based on the values of the following operational parameters of the Capture program:

- The **monitor_limit** parameter (on Linux, UNIX, Windows, z/OS) and the **MONLMT** parameter (on OS/400) that indicate how long rows remain in the IBMSNAP_CAPMON table
- The **trace_limit** parameter (on Linux, UNIX, Windows, z/OS) and the **TRCLMT** parameter (on OS/400) that indicate how long rows remain in the IBMSNAP_CAPTRACE table

Both the monitor limit and the trace limit parameters have a default value of one week. You can change these values depending on how long you need to preserve the historical Capture latency and throughput information in the IBMSNAP_CAPMON table and the auditing and troubleshooting data from the IBMSNAP_CAPTRACE table.

Pruning the signal table: The signal (IBMSNAP_SIGNAL) table is also pruned during each pruning cycle. A signal row is eligible for pruning if the SIGNAL_STATE column value is equal to `C`. A value of `C` indicates that the signal information is complete and is no longer required by the Capture program or for any user processing and is eligible for pruning. A signal row with a SIGNAL_TIME column value that is older than the current DB2 timestamp minus the retention limit parameter value is eligible for retention limit pruning.

Pruning other dynamic control tables

The Capture program performs pruning operations for only the tables that it maintains. The Apply program maintains consistent-change data (CCD) tables; therefore, the Capture program does not automatically prune these tables. Some types of CCD tables do not require pruning. Complete condensed CCD tables are updated in place.

The only records that you might want to remove from complete condensed CCD tables are those with an IBMSNAP_OPERATION column value of `D` (Delete) that have already been replicated to the dependent target tables. Non-condensed CCD tables contain historical data and can grow very large. Because you should preserve this data for auditing purposes, you should not perform pruning operations on non-condensed CCD tables.

You should, however, consider pruning your internal CCD tables. These tables can grow quickly if there is heavy update activity on your system. Only the most recent changes are fetched from internal CCD tables, so you do not need to retain the older rows.

To enable pruning for internal CCD tables, consider adding after-SQL statements to associated subscription sets to prune change data that has already been applied to all dependent targets. Alternatively, you can also add the necessary SQL DELETE statements to your automatic scheduling facilities to delete rows from these tables.

You should also manually prune the Apply trail (IBMSNAP_APPLYTRAIL) and Apply trace (IBMSNAP_APPLYTRACE) tables. If you define and use multiple

subscription sets with frequently run Apply programs, the IBMSNAP_APPLYTRAIL table grows rapidly and requires frequent pruning. The best way to manage the growth of these tables is to add an after-SQL statement or procedure call to one of your subscription sets. Alternatively, you can add an SQL DELETE statement to your automatic scheduling facilities.

Preventing replication failures and recovering from errors

This section describes methods to prevent and recover from replication failures that can affect your control tables and replication data:

- Preventing cold starts of the Capture program
- Recovering from I/O errors and connectivity failures on your control tables
- Retrieving lost source data

Preventing cold starts of the Capture program

You should perform a cold start of the Capture program only if you are starting the program for the first time or you need to refresh your control and target tables. If you cold start the Capture program, all of the target tables in your replication environment are refreshed.

When a Capture program starts with the warmns, warmsa, or warmsi option on Linux, UNIX, Windows, or z/OS, the program attempts to retrieve log records based on the restart point in the restart (IBMSNAP_RESTART) table. If the Capture program cannot find the log, the Capture warm start fails. If you started the Capture program using the warmns or warmsi option, the restart process terminates and issues an error message. If you started the Capture program using the warmsa option, the restart process stops and the Capture program performs a cold start, deleting all records in the CD and UOW tables.

To prevent a cold start of the Capture program, consider the following recommendations:

- On Linux, UNIX, Windows, and z/OS operating systems, specify the warmns or warmsi startmode instead of warmsa to restart a Capture program whenever possible. The warmns and warmsi options prevent an automatic cold start of the Capture program if the restart process fails. See “asncap: Starting Capture” on page 282 for more information.
- On OS/400 operating systems, start the capture program with the **RESTART(*YES)** parameter. The Capture program continues processing from the point where it was when it ended previously. See “STRDPRCAP: Starting Capture (OS/400)” on page 393 for more information.
- Use the Replication Alert Monitor or other mechanism to check the status of the historical data from your Capture programs. You can then use this information to verify that the Capture programs are always running if DB2 is active. See Chapter 12, “On-demand reporting for SQL replication,” on page 165 for more information.
- Make sure that you retain sufficient DB2 log data or journal receivers on your system and that this data is available to DB2 replication. See “Maintaining and retaining source logs and journal receivers” on page 205 for information regarding log retention.

Recovering from I/O errors and connectivity failures on your control tables

If you experience an I/O error or connectivity failure on any control table, use a standard DB2 recovery procedure to forward recover the table; the table will not lose any data.

If the Capture program detects an I/O error or connectivity failure, the program issues an appropriate error message and shuts down. After you correct the error, you can restart the Capture program from the point of failure.

The Apply program shuts down if it detects catastrophic errors on the control tables. If the Apply program detects errors on target tables or errors with network connectivity, the program writes the error to the Apply trail (IBMSNAP_APPLYTRAIL) table and then continues processing.

Retrieving lost source data

If a source table is forward recovered to the point of failure, DB2 replication proceeds normally. After the table is recovered, the Capture program continues collecting data changes for the table.

However, the Capture and Apply programs do not detect a point-in-time recovery of a read-only target table. If you recover a source table, the Apply program might have replicated changes to the target tables that no longer exist at the source, leaving inconsistencies between your source tables and target tables if you cannot take the target tables back to the same logical point in time.

This scenario becomes even more complex when there are multiple levels of replication. You must either develop a mechanism that provides matching recovery points among the various levels or use a full refresh as your recovery method of choice.

See "Coordinating replication events with database application events" on page 196 for more information about setting distributed recovery points.

Maintaining your target tables

Maintain the tables on the target server in the same way that you maintain other tables on your database system. Use your current backup and maintenance routines on these target tables, whether your target tables are existing database tables or tables that you specified to be automatically generated by DB2 replication.

Important: Deactivate your Apply programs before taking a target table offline to run any utility.

Related concepts:

- Chapter 23, "How the SQL replication components communicate," on page 415

Related tasks:

- Chapter 1, "Planning for SQL replication," on page 3
- Chapter 2, "Configuring servers for SQL replication," on page 15

Part 2. Replication Center

This part of the book contains the following chapters:

Chapter 15, "Using the Replication Center for SQL replication," on page 219 describes the Replication Center.

Chapter 16, "Basic SQL replication scenario: DB2 for Windows," on page 243 describes how to use the Replication Center to perform a simple replication scenario using sample data.

Chapter 15. Using the Replication Center for SQL replication

The Replication Center is a user interface tool that you can use to set up and administer your replication environment and to run the Capture, Apply, and Replication Alert Monitor programs. You can use the Replication Center to perform such administration tasks as:

- Create replication control tables
- Register replication sources
- Create subscription sets and add subscription-set members to the set
- Operate the Capture program
- Operate the Apply program
- Monitor the replication process

The Replication Center also has a launchpad that allows you to perform the basic functions needed to set up a DB2® replication environment. The launchpad shows you graphically how the different steps are related to one another.

You can use the Replication Center to set up DB2-to-DB2 replication environments or replication between DB2 and non-DB2 relational databases. The Replication Center is part of the DB2 Control Center set of tools. See the online help for detailed task information for the Replication Center.

This chapter helps you perform the following tasks:

Using the Replication Center

- See "Prerequisites for the Replication Center" on page 220. The prerequisites for the Replication Center are very similar to the prerequisites for the DB2 Control Center.
- See "Starting the Replication Center" on page 221. You can start the Replication Center several ways.
- See "Using the Replication Center launchpad for SQL replication" on page 222. Using the launchpad is optional, but can be very helpful for first-time users.

Setting up the Replication Center

- See "Managing user IDs and passwords for the Replication Center" on page 223. You can maintain the passwords that the Replication Center uses to connect to databases and log on to systems.
- See "Creating replication profiles" on page 224. Creating profiles is optional, but can be very helpful if you manage a large replication environment.
- See "Creating replication control tables" on page 227. You must create control tables in each database that will act as a replication control server.
- See "Adding servers to the Replication Center" on page 229. Servers are added to the Replication Center automatically when you create replication control tables. You can customize your view of your replication environment by adding only those servers that you want to administer.

- See “Enabling a database for change capture (UNIX and Windows)” on page 230. You must enable each Capture control server on Linux, UNIX® and Windows® systems for change capture and initiate a database backup.

Defining the replication environment

- See “Registering sources” on page 231. You can register tables or views as replication sources.
- See “Creating subscription sets” on page 232. You can create empty sets and add subscription-set members to them at any time, or you can create the subscription-set members as you create the subscription set.

Maintaining your replication environment

- See “Activating or deactivating subscription sets” on page 235. You can temporarily or permanently deactivate or activate any subscription set.
- See “Promoting replication objects” on page 236. You can promote table registrations and subscription sets from a test environment to a production environment.
- See “Forcing a full refresh of target tables” on page 237. You can control when the Apply program performs a full refresh for a subscription set
- See “Removing or deleting replication definitions” on page 238. You can remove replication objects from the Replication Center and you can delete replication definitions from a replication control server.

Operating your replication environment

- See “Operating the Capture program” on page 238. You can start and stop the Capture program on any server in your network. You can also perform many other operational tasks for the Capture program.
- See “Operating the Apply program” on page 239. You can start and stop the Apply program on any server in your network. You can also perform many other operational tasks for the Apply program.
- See “Operating the Replication Alert Monitor” on page 239. You can define alert conditions for monitoring replication activity.

Prerequisites for the Replication Center

Your system must have the correct Java™ Runtime Environment (JRE) installed to run the Replication Center. When you install DB2, you have the option to install the JRE. If you choose not to install the JRE, you must ensure that your system has Version 1.3 of either the Java 2 Runtime Environment or the Java 2 Software Development Kit.

To display z/OS™ buffer pools and to operate the Capture, Apply, or Replication Alert Monitor programs from the Replication Center, you must install the DB2 Administration Server for z/OS and the 390 Enablement package:

- DB2 for OS/390® and z/OS version 7 FMID for the DB2 Administration Server is HDAS810.
- DB2 for OS/390 and z/OS version 7 FMID for the 390 Enablement package is JDB771D. This package includes stored procedures that must be installed in DB2.
- DB2 for OS/390 version 6 FMID for the 390 Enablement package is JDB661D.

The Replication Center calls the DB2-supplied stored procedure DSNWZP to retrieve DB2 subsystem parameter information such as the SQL escape character.

The DSNWZP stored procedure is not part of the 390 Enablement package. Run the DB2 for z/OS installation DSNTIJSG job to define the DSNWZP stored procedure.

If you will use the Replication Center to operate the Capture, Apply, or Replication Alert Monitor programs on remote systems, ensure that the DB2 Administration Server (DAS) is running on the local system that is running the Replication Center and on each of the remote DB2 systems that will run the Capture or Apply programs. The DB2 Administration Server for z/OS is only available with DB2 for OS/390 and z/OS V7 or later. Once it is installed, it can also be used with DB2 for OS/390 V6 applications.

Configuring the Replication Center for host RDBMSs

If you are using the Replication Center or command-line tools to administer replication on a host database or subsystem, you must bind the Distributed Database Connection Services (DDCS) packages to the host DB2 UDB server on z/OS (MVS/ESA), VSE, VM, or AS/400®.

Prerequisites:

Before you bind the DDCS packages:

- Configure connectivity to the host database or subsystem.
- Catalog the host database or subsystem.

Procedure:

To bind DDCS packages to the host database or subsystem:

1. Go to the directory where the Capture program bind files are located. This directory is usually the \SQLLIB\BND directory on the drive where you installed DB2 UDB or the client application enabler (CAE). For example, to navigate to the default directory on Windows, issue the following command:
   ```
   cd C:\Program Files\IBM\SQLLIB\BND
   ```
2. Bind the packages to the host DB2 UDB for z/OS (MVS/ESA), VSE, VM, AS/400 database or subsystem by issuing the following commands:
   ```
   DB2 CONNECT TO dbname USER userid USING password
   DB2 BIND @DDCSxxx.LST ISOLATION CS BLOCKING ALL SQLERROR CONTINUE
   ```

   Where *dbname* is the name of the host database or subsystem, *xxx* specifies the platform of the host system (the choices for Replication are MVS™, VSE, VM, or AS/400), and CS specifies the cursor stability isolation level.

Starting the Replication Center

The Replication Center is installed as part of a typical DB2 installation for UNIX or Windows operating environments. If you perform a custom installation, you must select the General Administration Tools component to install the Replication Center.

To start the Replication Center, enter the **db2rc** command in a command window.

On Windows systems, you can also start the Replication Center by using the Windows **Start** menu:

1. Click **Start**.
2. Select **Programs**.

3. Select **IBM DB2**.
4. Select **General Administration Tools**.
5. Click **Replication Center**.

If you already have the DB2 Control Center running, you can start the Replication Center by selecting **Replication Center** from the **Tools** menu or by clicking the icon for the Replication Center.

Figure 7 shows the Replication Center.

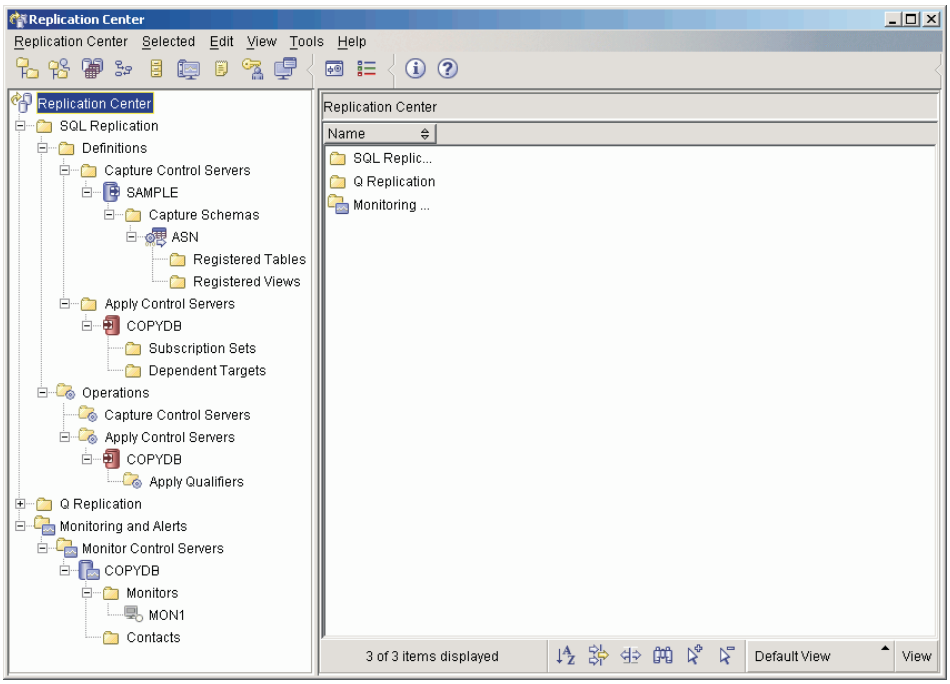


Figure 7. The Replication Center

Using the Replication Center launchpad for SQL replication

When you first start the Replication Center, you see the Replication Center launchpad. You can use the launchpad to perform the basic functions needed to set up a Q replication, event publishing, or SQL replication environment. It also shows you graphically how the different steps are related to one another.

The launchpad gives you central access to the most common functions in the Replication Center, but you can also access these functions from the object tree in the Replication Center. You can use the object tree to view or manipulate anything that you create using the launchpad. Many other, more advanced, functions are available in the Replication Center that are not accessible from the launchpad.

Note: For information on the event publishing and Q replication pages of the launchpad, see the *Replication and Event Publishing Guide and Reference*.

To access the SQL replication page, click the button next to the description of SQL replication. From the SQL replication page of the launchpad, you can perform the following tasks:

- Create the Capture control tables

This option opens the Create Capture Control Tables window, from which you can create the necessary replication control tables in a specific database for the Capture program.

- Register a source table

  This option opens the Register Tables window, from which you can define registration information (source columns, CD table information, and so on) for each source table that you want to register.

- Create the Apply control tables

  This option opens the Create Apply Control Tables window, from which you can create the necessary replication control tables in a specific database for the Apply program.

- Create a subscription set

  This option opens the Create Subscription Set window, from which you can define subscription set information: Capture, target, and Apply control servers, source-target mapping, properties for each subscription set member, schedule for the set, and SQL statements for the set.

- Start the Capture program

  This option opens the Start Capture window, from which you can start the Capture program and specify startup parameters for it.

- Start the Apply program

  This option opens the Start Apply window, from which you can start the Apply program and specify startup parameters for it.

Recommendation: The launchpad does not require that you perform these tasks in sequence, but if you are new to DB2 replication, follow the sequence that is presented on the launchpad. You can also skip or repeat steps in the launchpad if the necessary replication or database objects already exist. Use the launchpad or the object tree in the Replication Center to create the necessary replication and database objects.

You can use the launchpad at any time by selecting **Launchpad** from the **Replication Center** menu, or by right-clicking the Replication Center folder in the object tree and selecting **Start Launchpad**. If the Overview page is displayed, click **Getting Started with SQL Replication**. If another launchpad page is displayed, select **SQL replication launchpad** from the **Select launchpad view** field.

Managing user IDs and passwords for the Replication Center

The Replication Center must be able to connect to many database servers: source servers, Capture control servers, Apply control servers, Monitor control servers, and target servers. The Replication Center must also be able to connect to each system that runs the Capture program, the Apply program, or the Replication Alert Monitor. For all remote databases and systems, you need a valid user ID and password to connect to each database or to log on to each system. The Replication Center allows you to specify each user ID and password once, so that you are not prompted every time the Replication Center attempts to connect to a remote database or log on to each remote system.

By default, the Replication Center saves the user ID and password information in its metadata file. Because passwords are not encrypted in this file, you can specify that the Replication Center should not save passwords in this file but keep them only in memory. When you change a password in DB2, you must also change the password in the Replication Center so that they both use the same password. The

Replication Center does not share its password information with the Apply program, Replication Analyzer, or Replication Alert Monitor.

To manage user IDs and passwords for the Replication Center:

1. Right-click the **Replication Center** icon.
2. Select **Manage Passwords and Connectivity**.

   In the Manage Passwords and Connectivity window, you can perform any of the following tasks:

   - Add user connectivity information for servers or systems that you plan to use or are using in your replication environment.
   - Change connectivity information for servers or systems that you are using in your replication environment.
   - Remove connectivity information for servers or systems in your replication environment.
   - Test server or system connections by using the specified user ID and password.

   The Replication Center uses the connectivity information to perform the following actions:

   - Connect to local and remote servers to retrieve data and run SQL scripts.
   - Log on to remote systems to run commands.

Creating replication profiles

As part of replication setup, you create replication control tables, often in more than one database; you register many source tables and views, all of which have CD tables; and you define many target tables as part of creating subscription sets. For each of these replication objects, you probably have specific naming conventions and common attributes (for example, perhaps all of the table spaces for your CD tables use the same page size). The Replication Center allows you to create profiles that reflect these naming conventions and common definitions, rather than having to specify these common definitions every time you create replication objects. You can create profiles for the following replication objects:

- Replication control tables
- Replication source objects (CD tables)
- Replication target objects

In each object profile, you specify naming conventions for database objects such as CD tables, indexes, and table spaces, and you specify common attributes for these objects, such as page sizes and buffer pools. The values that you specify in each profile become the default values displayed in the Create Control Tables window, the Register Tables window, the Register Views window, or the Create Subscription Set window. You can override these default values when you create specific replication objects, or you can accept the values that you defined in your profile by clicking **OK**.

Creating control-table profiles

For each replication control table (for example, the register table, IBMSNAP_REGISTER), you can define table space information and index information in the profile. By default, the Replication Center groups the replication control tables together into table spaces for optimal performance. For these table spaces, you can define a naming convention that the Replication Center uses when it creates the table space, or you can specify a table space that already exists. You

can also define other operating-system specific table space information for the control tables. Most of the control tables also require one or more indexes. You can define a naming convention that the Replication Center uses when it creates these indexes, or you can specify an index that already exists.

You can define a unique control-table profile for each type of operating system that DB2 supports. You can also define profiles for each type of non-DB2 database that DB2 replication supports. The Replication Center does not provide a control-table profile for OS/400® systems because the replication control tables are created when you install DB2 DataPropagator™ for iSeries™.

To create control-table profiles:

1. Expand the **SQL Replication** folder.
2. Right-click the **Definitions** folder.
3. Select **Manage Control Table Profiles**.

   In the Manage Control Table Profiles window:

   a. Select the operating-system or non-DB2 database environment for which you are creating the profile.
   b. Select a replication control table from the list.
   c. Define the table space and index characteristics for the selected control table.

      Be sure to select every control table that creates a separate table space and define its characteristics.
   d. When you have defined all of the control tables for a particular operating-system platform or non-DB2 relational database system, click **Apply**. Click **Close** to close the Manage Control Table Profiles window.

Creating source-object profiles

For each DB2 replication source object (table or view), the Capture program requires a CD table. When you register a source object, you specify the name and characteristics for both the CD table and the index for the CD table. By creating a profile for source objects, you can define common characteristics for all sources that you register from a particular source database. Using these common characteristics, you can register many tables or views as part of a single action.

You can define a naming convention that the Replication Center uses when it creates the CD table, the table space for the CD table, and the index for the CD table. You can also define truncation rules for each of these objects if the name should exceed the operating-system-specific length limit (for example, 128 characters for UNIX and Windows databases) after the Replication Center applies the prefix and suffix that you specify. For example, you can create a profile that names your CD tables "CD_*sourcetablename*" (where *sourcetablename* varies for each registered source table) and places them in a table space named "CD_repltablespace".

You must add a Capture control server to the Replication Center before you can create a source-object profile for that server.

To create source-object profiles, use the Manage Source Object Profiles window. You can open this window in two ways:

- Open the window from the **Definitions** folder:
  1. Expand the **SQL Replication** folder.

2. Right-click the **Definitions** folder.
3. Select **Manage Source Object Profiles**.

   In the Manage Source Object Profiles window, select the source server for which you are creating the profile.

- Open the window from a source server:
  1. Expand the **SQL Replication** folder.
  2. Expand the **Definitions** folder.
  3. Expand the **Capture Control Servers** folder.
  4. Right-click a source server and select **Manage Source Object Profiles**.

     In this case, in the Manage Source Object Profiles window, you don't need to select the source server for which you are creating the profile.

In the Manage Source Object Profiles window, define the characteristics for the CD table, the table space for the CD table, and the index for the CD table. You can also select the truncation rules for each of these objects.

Creating target-object profiles

When you create a subscription-set member, you define a replication mapping between a source object (table, view, or nickname) and a target table. If the target table does not already exist, you specify the name and characteristics for both the target table and the index for the target table. By creating a profile for target objects, you can define common characteristics for all target tables in a particular target database.

You can define a naming convention that the Replication Center uses when it creates the target table, the table space for the target table, and index for the target table. You can also define truncation rules for each of these objects if the name should exceed the operating-system-specific length limit (for example, 128 characters for UNIX and Windows databases) after the Replication Center applies the prefix and suffix that you specify. For example, you can create a profile that names your target tables "TG_*sourcetablename*" (where *sourcetablename* varies for each registered source table) and places them in a table space named "TS_*targettablename*" (where *targettablename* varies for each target table).

You must catalog a target server in the local DB2 database before you can create a target-object profile for it. However, you do not need to add the target server to the Replication Center as a Capture control server, an Apply control server, or a Monitor control server.

To create target-object profiles:

1. Expand the **SQL Replication** folder.
2. Right-click the **Definitions** folder.
3. Select **Manage Target Object Profiles**.
4. In the Select Server window, select the database server for which you are creating a target-table profile and click **OK**.

   In the Manage Target Object Profiles window, select the target server for which you are creating the profile. Define the characteristics for the target table, the table space for the target table, and the index for the target table. You can also select the truncation rules for each of these objects.

Creating replication control tables

The replication control tables store all of the information about the setup for your replication environment, and they store operational information that the Capture and Apply programs use during replication. You must create replication control tables in a database before you can add that database server to the Replication Center. When you create replication control tables in a particular database, the Replication Center automatically adds that database server to the object tree in the Replication Center.

You cannot use the Replication Center to create replication control tables for OS/400 systems because the replication control tables are created when you install DB2 DataPropagator for iSeries. If you want to recreate the control tables, or create the control tables using an alternate Capture schema, use the OS/400 **CRTDPRTBL** command.

A Capture control server can also act as an Apply control server if you create all of the replication control tables in the same database. Likewise, a Capture or Apply control server can also act as a Monitor control server.

Creating Capture control tables

You can create control tables for a Capture control server in two ways:

- Open the Create Control Tables – Quick – Server Information window:
  1. Expand the **SQL Replication** folder.
  2. Expand the **Definitions** folder.
  3. Right-click the **Capture Control Servers** folder and select **Create Capture Control Tables → Quick**.
  4. In the Select a Server window, select the server in which you want to create Capture control tables and click **OK**.

     The Create Control Tables – Quick – Server Information window asks you some simple questions about your replication environment, and based on your answers, the Replication Center will create the replication control tables in specific table spaces of the appropriate sizes, and group the control tables in these table spaces for optimal performance.
- Open the Create Capture Control Tables window:
  1. Expand the **SQL Replication** folder.
  2. Expand the **Definitions** folder.
  3. Right-click the **Capture Control Servers** folder and select **Create Capture Control Tables → Custom**.
  4. In the Select a Server window, select the server in which you want to create Capture control tables and click **OK**. This server is where you will run the Capture program. If the database is a federated database that acts as a gateway for non-DB2 relational sources, select the federated database and retrieve the server mappings from that database to display a list of non-DB2 relational servers that are defined for that federated database.

     In the Create Capture Control Tables window, define the characteristics for each control table:

     a. Select a replication control table from the list.

     b. Define the table space and index characteristics for the selected control table.

        Be sure to select every control table that creates a separate table space and define the table space and index characteristics for that control table.

Tip: If you created a control-table profile for the operating-system platform of the selected database, you can accept the settings from your profile or override them.

You can specify a unique schema name for the Capture control tables; the default is ASN. A separate schema name is necessary if you plan to run more than one instance of the Capture program for the selected database.

c. When you have defined all of the control tables, click **OK**.

Creating Apply control tables

You can create control tables for an Apply control server in two ways:

- Open the Create Control Tables – Quick – Server Information window:
  1. Expand the **SQL Replication** folder.
  2. Expand the **Definitions** folder.
  3. Right-click the **Apply Control Servers** folder and select **Apply Capture Control Tables → Quick**.
  4. In the Select a Server window, select the server in which you want to create Apply control tables and click **OK**.

     The Create Control Tables – Quick – Server Information window asks you some simple questions about your replication environment, and based on your answers, the Replication Center will create the replication control tables in specific table spaces of the appropriate sizes, and group the control tables in these table spaces for optimal performance.
- Open the Create Apply Control Tables window:
  1. Expand the **SQL Replication** folder.
  2. Expand the **Definitions** folder.
  3. Right-click the **Apply Control Servers** folder and select **Apply Capture Control Tables → Custom**.
  4. In the Select a Server window, select the server in which you want to create Apply control tables and click **OK**.

     In the Create Apply Control Tables window, define the characteristics for each control table:

     a. Select a replication control table from the list.

     b. Define the table space and index characteristics for the selected control table.

        Be sure to select every control table that creates a separate table space and define the table space and index characteristics for that control table.

        Tip: If you created a control-table profile for the operating-system platform of the selected database, you can accept the settings from your profile or override them.

     c. When you have defined all of the control tables, click **OK**.

Creating Monitor control tables

You can create Monitor control tables in Linux, UNIX, Windows, VM/VSE, or z/OS databases. You should not create Monitor control tables in OS/400 databases or non-DB2 relational databases. You can use a Monitor control server to monitor replication activity for any DB2 database in your replication network, including OS/400 databases.

To create control tables for a Monitor control server:

1. Expand the **Monitoring and Alerts** folder.
2. Right-click the **Monitor Control Servers** folder and select **Create Monitor Control Tables**.
3. In the Select a Server window, select the server in which you want to create Monitor control tables and click **OK**.

   In the Create Monitor Control Tables window, define the characteristics for each control table:

   a. Select a replication control table from the list.
   b. Define the table space and index characteristics for the selected control table.

      Be sure to select every control table that creates a separate table space and define the table space and index characteristics for that control table.
   c. When you have defined all of the control tables, click **OK**.

Adding servers to the Replication Center

When you create replication control tables in a particular database, the Replication Center automatically adds that database server to the object tree in the Replication Center. You can also add or remove a database server from the object tree without affecting any of the replication objects that you created in that database and without affecting the Capture program, the Capture triggers, the Apply program, or the Replication Alert Monitor that might be running on that server. The Replication Center does not automatically list all of your locally cataloged databases for the following reasons:

- The Replication Center shows only valid replication objects. If a locally cataloged database does not contain replication control tables, Replication Center does not display that database in the object tree.
- If your replication environment restricts the authority for creating replication control tables to an administrator, you can still allow other people to manage replication objects (for example, registered sources or subscription sets) in databases of interest to them.
- Even if everyone in your replication team has the same authority, each person might want to focus only on certain replication servers. Each person can add just those database servers that he or she wants to administer, even if your replication environment includes more than the Replication Center shows.

Important: Before you can add a database server to the Replication Center, you must first catalog the server in the local DB2 database and ensure that replication control tables exist in the database.

You can add the following servers to the Replication Center:

- Capture control server

  To add a Capture control server to the Replication Center:

  1. Right-click the **Capture Control Servers** folder and select **Add**. The Add Capture Control Server Wizard opens.
  2. After reading the Getting started page, click **Next** to display the Specify a Capture control server to add to the Replication Center page.
  3. Click the push button on the **Server alias** field. The Select a Server window opens.
  4. Select the server you want to add to the Replication Center and click **OK**. The Select a Server window closes.

5. Type the User ID and password for the server in the **User ID** and **Password** fields.
6. Click **Next** to display the Summary page.
7. If you are satisfied with the information, click **Finish**. If the information is incorrect, click **Back** and make the necessary changes.

You can also add non-DB2 relational servers to the Replication Center as Capture servers by right-clicking a particular federated database displayed in the Add Capture Control Servers window and selecting **Retrieve non-DB2 Server(s)**. The Replication Center adds the non-DB2 relational server defined for that federated database to the table.

- Apply control server

  To add a Apply control server to the Replication Center:

  1. Right-click the **Apply Control Servers** folder and select **Add**. The Add Apply Control Server Wizard opens.
  2. After reading the Getting started page, click **Next** to display the Specify an Apply control server to add to the Replication Center page.
  3. Click the push button on the **Server alias** field. The Select a Server window opens.
  4. Select the server you want to add to the Replication Center and click **OK**. The Select a Server window closes.
  5. Type the User ID and password for the server in the **User ID** and **Password** fields.
  6. Click **Next** to display the Summary page.
  7. If you are satisfied with the information, click **Finish**. If the information is incorrect, click **Back** and make the necessary changes.

- Monitor control server

  To add a Monitor control server to the Replication Center:

  1. Right-click the **Monitor Control Servers** folder and select **Add**. The Add Monitor Control Server Wizard opens.
  2. After reading the Getting started page, click **Next** to display the Specify a Monitor control server to add to the Replication Center page.
  3. Click the push button on the **Server alias** field. The Select a Server window opens.
  4. Select the server you want to add to the Replication Center and click **OK**. The Select a Server window closes.
  5. Type the User ID and password for the server in the **User ID** and **Password** fields.
  6. Click **Next** to display the Summary page.
  7. If you are satisfied with the information, click **Finish**. If the information is incorrect, click **Back** and make the necessary changes.

Enabling a database for change capture (UNIX and Windows)

The default logging for a DB2 database on UNIX or Windows systems is circular logging, which uses a fixed-size file that is reused when the log fills. Replication requires archival logging, which uses one or more log files that grow indefinitely and are never reused (of course, you can use DB2 utilities to manage an archive log to ensure that it doesn't fill all your disk space).

To enable archival logging for DB2 databases:

1. Expand the **SQL Replication** folder.
2. Expand the **Definitions** folder.
3. Expand the **Capture Control Servers** folder.
4. Right-click the database for which you want to enable archival logging, and select **Enable Database for Replication**.
5. Click **OK** on the Enable Database for Replication window to change the database configuration (to set LOGRETAIN to RECOVERY) and to initiate a database backup.

   If you also want to use an exit routine to manage archived logs, you must manually set the USEREXIT database-configuration parameter.

You can also select several databases in the contents pane for the **Capture Control Servers** folder and enable archival logging for all selected databases at the same time.

You do not need to enable archival logging for databases on other operating-system platforms because the default logging for those environments is archival. You also do not need to enable archival logging for non-DB2 relational databases because the Capture triggers do not depend on the database logs.

Registering sources

To register one or more tables for replication:

1. Expand the **Capture Control Servers** folder.
2. Expand the database server that contains the source tables that you want to register.
3. Expand the **Capture Schemas** folder.
4. Expand the schema that contains the source tables that you want to register.
5. Right-click the **Registered Tables** folder and select **Register Tables**. The Add Registrable Tables window opens.

   Because the database could contain many hundreds of tables, you can prefilter the list of tables so that the Register Tables window shows only those tables that you are interested in.
6. From the Add Registrable Tables window, specify the search criteria, if any, and click **Retrieve**. If you want to include all tables, click **Retrieve All**.
7. Select one or more tables from the filtered list that you want to register as a replication source and click **OK**. The Register Tables window remains open.
8. From the **Selected tables** list, select the first table that you want to register as a replication source. You can define the following information for a replication source:
   - The row-capture rule that specifies when the Capture program writes a row to the CD table (or when the Capture triggers write a row to the consistent-change data (CCD) table).
   - The specific columns that you want to make available for replication, including before-image and after-image columns.

     Any column that you do not register will not be available for any subscription set.

     Recommendation: Consider the potential target tables that you might define using this table as a source. If the key columns for the target table can be updated at the source, register the before-image values of the columns at the source that will make up the key columns at the target. When you define

which targets subscribe to this source (by creating subscription sets), you can use the Apply program to perform special updates to the target key columns by using these before-image values.

Any column that you do not register will not be available for any subscription set.

- The prefix to use for registered before-image columns to associate them with the after-image columns.
- Whether you want to allow the Apply program to refresh target tables based on this source table.
- Whether to capture changes as delete and insert pairs (useful for changes to partitioning keys).
- Whether changes are recaptured in dependent replicas in an update-anywhere scenario.
- The level of conflict detection for update-anywhere scenarios.

  For peer-to-peer scenarios, you must select **No detection**.

For more information about these options, see the online help for the Replication Center.

For each registered source table, you also specify information about the CD table and the index for the CD table. If you created a source-object profile for this database server, you can accept the defaults that you defined in the profile, or you can override them.

To register a view, right-click the **Registered Views** folder and select **Register Views**. The view must already exist before you can register it as a replication source. If it does not exist, click **Create View** in the Register Views window. In the Create View window, specify the view name and the SQL statement that defines the view. You can click **SQL Assist** to use the SQL Assist window to create the SQL statement that defines the view.

You can register an OS/400 table that is journaled remotely in the same way as you register any table, except that you must specify the source-table name as well as the journal library and journal-receiver name. However, you do not need to specify the journal library and journal name if they are the same as the journal library and journal name used by the source table or file.

You can register a nickname in the same way as you register a table, except that you must specify the nickname for the table (stored in DB2) instead of the actual table name (stored in the non-DB2 database).

Creating subscription sets

After you register one or more source tables, nicknames, or views, you need to subscribe to those sources; that is, to create a subscription set and add members to the set. You can create an empty subscription set and add members to it later, or you can add all the members while you create the subscription set.

To create a subscription set:

1. Expand the **Capture Control Servers** folder.
2. Expand the database server that contains the source tables for which you want to create subscription sets.
3. Expand the **Capture Schemas** folder.

4. Expand the schema that contains the source tables for which you want to create subscription sets.
5. Click the **Registered Tables** folder.
6. In the contents pane for the **Registered Tables** folder, right-click a source table and select **Create Subscription Set**. The Create Subscription Set window opens.

As an alternative, you can also create a subscription set using the following steps:

1. Expand the **Apply Control Servers** folder.
2. Expand a specific Apply control server.
3. Right-click the **Subscription Sets** folder and select **Create**. The Create Subscription Set window opens.

Creating a subscription set includes four major subtasks:

- "Defining the information for the subscription set."
- "Mapping sources to targets."
- "Scheduling the subscription set" on page 235.
- "Adding SQL statements or stored procedures to the subscription set" on page 235.

After you create the subscription set, you can edit the subscription set, add or remove subscription-set members, add or remove statements or procedures, activate the subscription set, force a full refresh of its members, or promote it to another database.

For more information about creating subscription sets, see the online help for the Replication Center.

Defining the information for the subscription set

In the Create Subscription Set window, you can define the following information for the subscription set:

- The alias for the Apply control server
- The subscription-set name
- The Apply qualifier
- The alias for the Capture control server
- The Capture schema that identifies the set of Capture control tables that define the registered sources for the subscription set
- The alias for the target server
- Whether the subscription set should be active as soon as it is created

  By default, a new subscription set is deactivated it as soon as it is created. You can choose to make it eligible to be processed by the Apply program immediately, or you can choose to activate it for just one Apply cycle.
- The subscription-set processing properties

Mapping sources to targets

After you define the subscription-set information, you can map the source tables and views to the target tables. On the Source-to-Target Mapping page of the Create Subscription Set window:

1. Click **Add** to display the Add Registered Sources window. From this window, you can filter the list of registered sources for the selected source database.

2. Select one or more tables from the filtered list that you want to add as the source for the subscription-set member and click **OK**. The Create Subscription Set window remains open at the Source-to-Target Mapping page.
3. In the table on the Source-to-Target Mapping page, select the target schema, name, or target type to change any of these values for target tables that do not already exist. After you choose a source, the target schema and target name are automatically generated based on the target-object profile for the selected target server, if one exists.
4. In the table on the Source-to-Target Mapping page, select a source-target pair in the table and click **Change** to display the Member Properties window. From this window, you can specify the exact mapping between a source table and a target table, including:
   - Selecting to which source columns the target will subscribe
   - Mapping source columns to target columns, including creating calculated columns
   - Specifying the index for the target table

     Important: If the columns that make up the index of the target can be updated at the source, select the target key change option. This option tells the Apply program to make special updates to the target key columns whenever the target key changes. The Apply program uses the before-image values of the columns at the source to make these special updates to the target. If you did not define the source registration to capture the before-image values of the columns that make up the key at the target, then you must alter your registration to include these before-image values before selecting the target-key change option.
   - Optionally, filtering the source rows with a WHERE clause, so that the target table includes only a subset of the source data
   - For Linux, UNIX, Windows, and z/OS systems, specifying the table space for the target table

   For replica target types you also specify the replica definition (row-capture rule, whether to recapture changes, and how to handle updates), the CD table for the replica table, and the index for the CD table.

   For CCD tables you also specify the properties of the CCD table, including whether it is complete or noncomplete, condensed or noncondensed, and whether you want to register it as a replication source.

If you want to create an empty subscription set, leave the Source-to-Target Mapping page empty. You can add subscription-set members to the subscription set later. You can add members to an existing subscription set by using one of the following notebooks:

- Subscription Set Properties. Use this notebook if you have already created the subscription set and want to add one or more subscription-set members to it.

  From the contents pane for the **Subscription Sets** folder, right-click a subscription set and select **Properties**.
- Add Members to Subscription Sets. Use this notebook to add one member to multiple subscription sets. For example, if you select four subscription sets when you open this notebook, you can add one member to each. Each member must use the same source.

  In the contents pane for the **Registered Tables** folder, right-click a source table and select **Add Member**.

  You can add a registered source to multiple subscription sets by using the Add Members to Subscription Sets notebook. Thus, you can create several empty

subscription sets and populate them all with the same source-target mappings. All of the subscription sets selected for the Add Members to Subscription Sets notebook must use the same Capture server and Capture schema.

Scheduling the subscription set

After you map sources to targets (or create an empty subscription set), you define subscription-set timing information. On the Schedule page of the Create Subscription Set window, specify when the subscription set should first be eligible for processing; the default is the current date and time of the local machine. You also specify the timing for how often the subscription set should be eligible for processing:

- Time-based replication

  The Apply program will process this subscription set using a regular time interval.
- Event-based replication

  The Apply program will process this subscription set whenever an event occurs.
- Both time-based and event-based replication

  The Apply program will process this subscription set using both a regular time interval and whenever an event occurs. In this case, the subscription set will be eligible for processing at both the scheduled time and when the event occurs.

Adding SQL statements or stored procedures to the subscription set

After you define subscription-set timing information, you can optionally add SQL statements or stored procedures to the subscription set. On the Statements page of the Create Subscription Set window, you can add SQL statements or stored procedures that the Apply program should run while processing the subscription set. Click **Add** to add a statement or procedure to the subscription set.

In the Add SQL Statement or Procedure Call window, you can enter an SQL statement or use SQL Assist to define the statement. You can specify that these statements or procedures should run at the target server, before or after processing the subscription set, or at the Capture control server before processing the subscription set. You can also add SQLSTATE values that the Apply program will accept as successful, such as `02000` when attempting to delete a row that does not exist. Because these SQLSTATE values are treated as successful, the error condition does not appear in the Apply trail table (IBMSNAP_APPLYTRAIL), and the Replication Alert Monitor does not generate an alert for them.

Activating or deactivating subscription sets

Usually, you want your subscription sets to be active so that they can be processed by the Apply program. However, there are times when you might want to deactivate a subscription set for a short time, or indefinitely. If you deactivated a subscription set when you created it, you will probably want to activate it at some time.

To deactivate an active subscription set:

1. Click the **Subscription Sets** folder to display the subscription sets in the contents pane.
2. Right-click an active subscription set, and select **Deactivate**. The Replication Center deactivates the subscription set immediately.

To activate an inactive subscription set:

1. Click the **Subscription Sets** folder to display the subscription sets in the contents pane.
2. Right-click an inactive subscription set, and select one of the following options:
   - **Activate → Indefinitely** to activate the subscription set.
   - **Activate → One-time only** to activate the subscription set for only one Apply cycle.

   The Replication Center activates the subscription set immediately.

Promoting replication objects

After you register sources and create subscription sets on a database server, you might want to copy the replication definitions to another database (for example, from a test system to a production system) without having to re-register the sources or re-create the subscription sets. The Replication Center provides Promote functions to help you copy replication definitions from one database to another.

Restrictions:

- You can use the Promote functions to copy replication definitions only between *like* systems, for example from one DB2 for UNIX and Windows system to another DB2 for UNIX and Windows system, but not from a DB2 for UNIX and Windows system to a DB2 for z/OS system. As long as the systems are all the same type of operating-system platform, you can use the Promote functions for OS/400, Linux, UNIX, Windows, or z/OS systems.
- You cannot use the Promote functions to copy replication definitions for non-DB2 databases or federated database objects.
- You cannot use the Promote functions to copy replication definitions that include OS/400 remote journals.

Promoting registered tables or views

To promote registered tables:

1. Click the **Registered Tables** folder to display the registered source tables in the contents pane.
2. Right-click a source table and select **Promote**. The Promote Registered Tables window opens.
3. In the Promote Registered Tables window, specify the information for the database server to which you want to copy the registration information:
   - Capture control server alias

     Select the new Capture control server for the registered source table.
   - Capture schema

     Specify the new Capture schema for the registered source table.
   - CD table schema

     Specify the new schema name for the CD table that is associated with the source table.
   - Table schema

     Specify the new schema name for the table. You can use the Promote function to create the source table in the new database.

To promote registered views, click the **Registered Views** folder to display the registered source views in the contents pane, right-click a source view and select **Promote**.

Promoting subscription sets

To promote subscription sets:

1. Click the **Subscription Sets** folder to display the subscription sets in the contents pane.
2. Right-click a subscription set and select **Promote**. The Promote Subscription Set window opens.
3. In the Promote Subscription Set window, specify the information for the database server to which you want to copy the subscription-set information:
   - Apply control server alias

     Select the new Apply control server for the subscription set. You can select the Apply control server that is already defined for the subscription set.
   - Capture control server alias

     Select the new Capture server for the subscription set. You can select the Capture control server that is already defined for the subscription set.
   - Target server alias

     Select the new target server for the subscription set. You can select the target server that is already defined for the subscription set.
   - Apply qualifier

     Type a new Apply qualifier for the subscription set.
   - Subscription set name

     Type a new name for the subscription set.
   - Capture schema

     Type a new Capture schema for the source tables in the subscription set.
   - Schema name for the source table or view

     Type a new schema name for the source tables in the subscription set.
   - Schema name for the target table or view

     Type a new schema name for the target tables in the subscription set.

   You can leave any field blank if you want to use the value from the current subscription-set definition.

Forcing a full refresh of target tables

There are times when you might need to reload a target table. For example, a gap in the source database log or journal might cause the Capture program to stop and request a cold start, which requires a full refresh of all target tables based on that source database. For small tables, you can let the Apply program perform the full refresh automatically. For large tables, you should use the ASNLOAD exit routine.

Using the Replication Center, you can bypass the full refresh that is normally done by the Apply program so that you can perform an unload or extract from the source table and a load to the target table. The Replication Center makes the necessary changes to the replication control tables to ensure that replication continues to run after the load is complete.

To perform a manual full refresh:

1. Click the **Subscription Sets** folder to display the subscription sets in the contents pane.
2. Right-click a subscription set and select **Full Refresh → Manual**.
3. Read the text in the Full Refresh – Manual Introduction window and click **Next**.
4. Click **Next** in the subsequent windows until you have completed the task.

The Full Refresh – Manual window helps you perform the following steps:

1. Disable current subscriptions for the selected subscription sets.

   After the subscription sets are disabled, you can unload the source table and load the target table.
2. Re-enable the subscriptions for the selected subscription sets.

You can run the generated SQL scripts for steps 1 and 2 immediately, or at a later time. Be sure to perform these steps in the order suggested by the Full Refresh – Manual window, or your replication environment might produce unpredictable results.

To perform an automatic full refresh, so that the Apply program will initiate the full refresh during the next Apply cycle:

1. Click the **Subscription Sets** folder to display the subscription sets in the contents pane.
2. Right-click a subscription set and select **Full Refresh → Automatic**.

Removing or deleting replication definitions

You can use the Replication Center to remove or delete any of the replication definitions that you create. You can perform any of the following tasks:

- Remove user IDs from the Replication Center
- Drop replication control tables from a Capture control server, an Apply control server, or a Monitor control server (not applicable for an OS/400 system)
- Remove a Capture control server, an Apply control server, or a Monitor control server from the Replication Center
- Delete registrations for source tables or views
- Delete subscription sets
- Delete members from subscription sets
- Delete statements from subscription sets
- Remove stored procedures from subscription sets

See the Replication Center online help for information about each of these tasks.

Operating the Capture program

You can perform many daily operations tasks for replication from the Replication Center. For example, you can start or stop the Capture program. To operate the Capture program, expand the **Operations** folder, then click the **Capture Control Servers** folder to display the currently defined Capture control servers in the contents pane. Right-click one of the Capture control servers, and select one of the following operational tasks:

- Start the Capture program
- Stop the Capture program
- Suspend the Capture program

- Resume the Capture program (after suspending it)
- Initiate the Capture pruning process to prune the following tables: CD, UOW, Capture monitor, Capture trace, and signal
- Reinitialize the Capture program so that it re-reads the register table
- View or change the values stored in the Capture parameters table
- View or change the current parameters that the Capture program is using
- View messages issued by the Capture program
- View statistics gathered by the Capture program:
  - The number of rows that the Capture program inserted into the CD table or skipped
  - The number of rows that the Capture program pruned from CD tables
  - The number of transactions that the Capture program committed
  - How much memory the Capture program is using
- View the average latency of the Capture program
- Query the status of the Capture program

You can perform any of these tasks for a Capture program that is running anywhere in your replication network.

Operating the Apply program

You can also use the Replication Center to operate the Apply program. To operate the Apply program, expand the **Operations** folder, click the **Apply Control Servers** folder, expand one of the Apply control servers, and click the **Apply Qualifiers** folder to display currently defined Apply qualifiers in the contents pane. Right-click one of the Apply qualifiers, and select one of the following operational tasks:

- Start the Apply program
- Stop the Apply program
- Display a report for subscription-set activity:
  - Display all subscription sets
  - Display failed subscription sets
  - Display successful subscription sets
  - Display an error summary report for each failed subscription set
- Display performance information for the Apply program:
  - Display the number of rows fetched from CD tables
  - Display the elapsed time for each subscription set
- Display a report on the end-to-end latency for each subscription set
- Query the status of the Apply program

You can perform any of these tasks for an Apply program that is running anywhere in your replication network.

Operating the Replication Alert Monitor

You can use the Replication Center to define contacts and alert conditions for the Replication Alert Monitor.

To create contacts that will be notified when the Replication Alert Monitor detects any of the specified alert conditions:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand a Monitor control server.
4. Right-click the **Contacts** folder and select **Create Contact → Person** or **Create Contact → Group**.
5. In the Create Contact window, specify the person's name and e-mail or pager address. In the Create Contact Group window, specify the name of the group and the members of the group.

To select alert conditions for the Capture program:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand a Monitor control server.
4. Right-click the **Monitors** folder and select **Create**. The Create Monitor Wizard opens.
5. On the Start page, specify the Monitor qualifier that is associated with the instance of the Replication Alert Monitor.
6. Under **SQL Replication**, select the **Capture programs** check box.
7. Click **Next** to display the Select alert conditions for Capture programs page.
8. Click **Add**. The Select alert conditions for Capture schemas window opens.
9. Specify the following information:
   - The Capture control server that you want to monitor
   - The Capture schema that you want to monitor
   - Any alert conditions
10. Click **OK** to close the window.
11. Click **Next** to display the Summary page.
12. If you are satisfied with the information, click **Finish**. If the information is incorrect, click **Back** and make the necessary changes.

To select alert conditions for the Apply program by Apply qualifiers:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand a Monitor control server.
4. Right-click the **Monitors** folder and select **Create**. The Create Monitor Wizard opens.
5. On the Start page, specify the Monitor qualifier that is associated with the instance of the Replication Alert Monitor.
6. Under **SQL Replication**, select the **Apply programs by Apply qualifier** check box.
7. Click **Next** until the Select alert conditions for Apply qualifiers, including all subscription sets that they process page is displayed.
8. Click **Add**. The Select alert conditions for Apply qualifiers window opens.
9. Specify the following information:
   - The Apply control server that you want to monitor
   - The Apply qualifier that you want to monitor
   - Any alert conditions
10. Click **OK** to close the window.
11. Click **Next** to display the Summary page.

12. If you are satisfied with the information, click **Finish**. If the information is incorrect, click **Back** and make the necessary changes.

To select alert conditions for the Apply program by subscription sets:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand a Monitor control server.
4. Right-click the **Monitors** folder and select **Create**. The Create Monitor Wizard opens.
5. On the Start page, specify the Monitor qualifier that is associated with the instance of the Replication Alert Monitor.
6. Under **SQL Replication**, select the **Apply programs by subscription sets** check box.
7. Click **Next** to display the Select alert conditions for individual subscription sets page.
8. Click **Add**. The Select alert conditions for subscription sets window opens.
9. Specify the following information:
   - The Apply control server that you want to monitor
   - The subscription set that you want to monitor
   - Any alert conditions
10. Click **OK** to close the window.
11. Click **Next** to display the Summary page.
12. If you are satisfied with the information, click **Finish**. If the information is incorrect, click **Back** and make the necessary changes.

To start the Replication Alert Monitor for a monitor qualifier:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand a Monitor control server.
4. Expand the **Monitor** folder.
5. Right-click a monitor qualifier and select **Start Monitor**.

You can perform any of these tasks for a Replication Alert Monitor that is running anywhere in your replication network.

Chapter 16. Basic SQL replication scenario: DB2 for Windows

Use the scenario in this chapter to gain some experience using the Replication Center and the Capture and Apply programs. Follow the steps in this simple scenario to copy changes from a DB2® replication source to a target table in a database on DB2 for Windows® Enterprise Server Edition (ESE) or Workgroup Server Edition (WSE).

The scenario consists of the following parts:

1. "Before you begin"
2. "Planning this scenario" on page 244
3. "Setting up the replication environment for this scenario" on page 246
4. "Operating in a replication environment" on page 256

Before you begin

If you want to work through this scenario on your computer, set up your system using these steps:

1. Make sure that you have DB2 for Windows installed on your computer.
2. Make sure you have created the default DB2 instance. This scenario assumes that all databases are within the same instance.
3. Make sure you have access to the SAMPLE database. This database will be both the source server and the Capture control server for this scenario.

   To create the SAMPLE database, use the **First Steps** application: select **Start → Programs → IBM DB2 → Set-up Tools → First Steps**). After the database is created, close the First Steps window.

   If you did not install First Steps when you installed DB2, open a DB2 command window and issue the **db2sampl** command to create the SAMPLE database.
4. Use the DB2 Control Center to create a new database called COPYDB, which you will use as the target server and the Apply control server. To create the database, right-click the **All Databases** folder, select **Create → Database Using Default**, and follow the instructions for creating a new database with default options. Both the name and the alias for the database should be COPYDB.

The steps in this chapter use the data in the DEPARTMENT table from the SAMPLE database. The fully qualified name is *schema*.DEPARTMENT; where *schema* is the user ID that created the table. Table 20 shows the DEPARTMENT table.

Table 20. The DEPARTMENT table

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	-
B01	PLANNING	000020	A00	-
C01	INFORMATION CENTER	000030	A00	-
D01	DEVELOPMENT CENTER	-	A00	-

Table 20. The DEPARTMENT table (continued)

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
D11	MANUFACTURING SYSTEMS	000060	D01	-
D21	ADMINISTRATION SYSTEMS	000070	D01	-
E01	SUPPORT SERVICES	000050	A00	-
E11	OPERATIONS	000090	E01	-
E21	SOFTWARE SUPPORT	000100	E01	-

For the remainder of this scenario, use the user ID with which you created the SAMPLE and COPYDB databases. Because you created the databases, you have the required authority (DBADM or SYSADM) to perform replication tasks.

Planning this scenario

Assume that your group uses an application that generates reports. This application needs information that exists in the DEPARTMENT table of the SAMPLE database. Instead of using the data directly from the source table, you want to copy the changes to a target table that can be read only by the report-generating application. For ease of administration, you want to keep the target table on the same machine as the source table.

You require a simple data distribution configuration, with changes from one replication source being replicated to a single read-only copy. This section describes the design and planning issues that you need to consider before you perform any replication tasks.

Replication source

You already know that the replication source is the *schema*.DEPARTMENT table in the SAMPLE database. Before you set up your environment, you must decide what you want to replicate from that table; you decide to register all columns and subscribe to all columns.

Replication target

You decide that you want your replication target to be the COPYDB database, which you created in "Before you begin" on page 243. Currently there is no target table in that database; you want the Replication Center to create the target table according to your specifications. This method of automatically generating a target table is preferred because it ensures correct mapping to the replication source. You can instead use existing target tables, but this scenario assumes that the target table does not exist.

Assume that you want the target table in COPYDB to contain the columns of information shown in Table 21.

Table 21. Columns for the COPYDB table

Column	Description
DEPTNO	Information from the DEPTNO column in the replication source table. This column will be the primary key of the target table.
DEPTNAME	Information from the DEPTNAME column in the replication source table.

Table 21. Columns for the COPYDB table (continued)

Column	Description
MGRNO	Information from the MGRNO column in the replication source table.
ADMRDEPT	Information from the ADMRDEPT column in the replication source table.
LOCATION	Information from the LOCATION column in the replication source table.

Because the columns in the target table simply reflect the data from the source table, and because there will be only one row in the target table for each row in the source table, you can use a *user copy* type of target table.

Replication options

For the purpose of this scenario, you decide to store the CD table, the target table, and the replication control tables in their respective default table spaces, as shown in Table 22. Although the SAMPLE and COPYDB databases exist in the same machine, their table spaces are in separate containers.

Table 22. Tables and table spaces used in this scenario

Database	Tables	Table spaces	Contents
SAMPLE	*schema*.DEPARTMENT	USERSPACE1	Source table
	schema.CDDEPARTMENT	TSCDDEPARTMENT	The CD table for the DEPARTMENT table
	Capture control tables	TSASNCA and TSASNUOW	The replication control tables for the Capture program
COPYDB	*schema*.TGDEPTCOPY	TSTGDEPTCOPY	Target table
	Apply control tables	TSASNAA	The replication control tables for the Apply program
	Monitor control tables	REPLMONTS1, REPLMONTS2, and REPLMONTS3	The replication control tables for the Replication Alert Monitor

Typically, you should create the CD table in a separate table space from the source table to reduce potential contention at the table-space level. You should accept the defaults (or define a profile within the Replication Center) for the table spaces of the replication control tables. For a production environment, it is best if you create each table space on a separate device, to reduce potential contention.

For scheduling replication, assume that you want DB2 replication to check for any changes from the source table every minute and replicate them to the target table. Although a report-generating application doesn't require that kind of response time, you want to test the replication environment to make sure that everything is working correctly.

Also, you decide that after each replication cycle, you want to delete any records from the Apply trail table that are older than one week (seven days). This pruning prevents the table from growing too large.

Setting up the replication environment for this scenario

After planning the replication model, you are ready to set up the replication environment. Because almost all of the following steps use the Replication Center, be sure that it is running: from the Windows **Start** menu: select **Programs → IBM DB2 → General Administration Tools → Replication Center**.

Step 1: Create replication control tables for the Capture program

The Capture program reads the replication control tables for current registration information and stores its status in these tables. Any database that will act as a Capture control server must contain the Capture control tables.

To create Capture control tables:

1. Expand the **SQL Replication** folder.
2. Expand the **Definitions** folder.
3. Right-click the **Capture Control Servers** folder and select **Create Capture Control Tables → Quick**. Alternatively, you could customize the Capture control tables by selecting **Create Capture Control Tables → Custom**.
4. In the Select a Server window, select the SAMPLE database. This database will be your Capture control server. Click **OK**.
5. In the Create Control Tables - Quick - Server Information window, select **Host sources for replication and capture changes to those sources**. Then click **Next**.
6. In the Create Control Tables - Quick - Replication Details window, click **Next**. You do not need to change any of the information in this window.
7. In the Create Control Tables - Quick - Table Spaces window, enter the table space specification for the TSASNCA table space. For example, set the buffer pool to IBMDEFAULTBP. For this scenario, accept the default Capture schema, ASN.
8. In the Create Control Tables - Quick - Table Spaces window, enter the table space specification for the TSASNUOW table space.
9. After you enter information for both table spaces in the Create Control Tables - Quick - Table Spaces window, click **OK**.
10. Click **Close** on the Message Dialog window. This window shows the result of generating the SQL script that will create the Capture control tables. If there were any errors, they would be displayed in this window.
11. Enter a valid user ID and password in the Run Now or Save SQL window and click **OK** to run the SQL script immediately.
12. You should see a message in the DB2 Message window that the script ran successfully. Click **Close**.
13. Expand the **Capture Control Servers** folder. The SAMPLE database should be displayed under the folder.

Step 2: Enable the source database for replication

The Capture program reads the DB2 log for log records that include changes to registered tables. The log must be an archive log so that the log file will not be reused by DB2 before the Capture program can read the log. For UNIX® and Windows environments, the DB2 default is circular logging, so you must change this setting to archive logging.

To enable the source database for replication:

1. Expand the **Capture Control Servers** folder.
2. Right-click the SAMPLE database and select **Enable Database for Replication**.
3. Click **OK** on the Enable Database for Replication window to use archive logging for the SAMPLE database and to initiate a backup for the database.
4. In the Backup window, specify the information for the database backup, and click **Backup Now**.

After you backup the database, you could start the Capture program, but do not start it yet. If you want to start the Capture program, see "Step 7: Replicate the scenario data" on page 255.

Step 3: Register a replication source

After you create the Capture control tables and enable the database for replication, register the DEPARTMENT table as a replication source.

To register a table as a replication source:

1. Expand the **SQL Replication** folder
2. Expand the **Definitions** folder.
3. Expand the **Capture Control Servers** folder.
4. Expand the SAMPLE database.
5. Expand the **Capture Schemas** folder.
6. Expand the ASN schema.
7. Right-click the **Registered Tables** folder and select **Register Tables**.
8. In the Add Registerable Tables window, click **Retrieve All** to list all the tables in the SAMPLE database that you can register as replication sources. Select the DEPARTMENT table and click **OK**. The Register Tables window is displayed, as shown in Figure 8 on page 248.

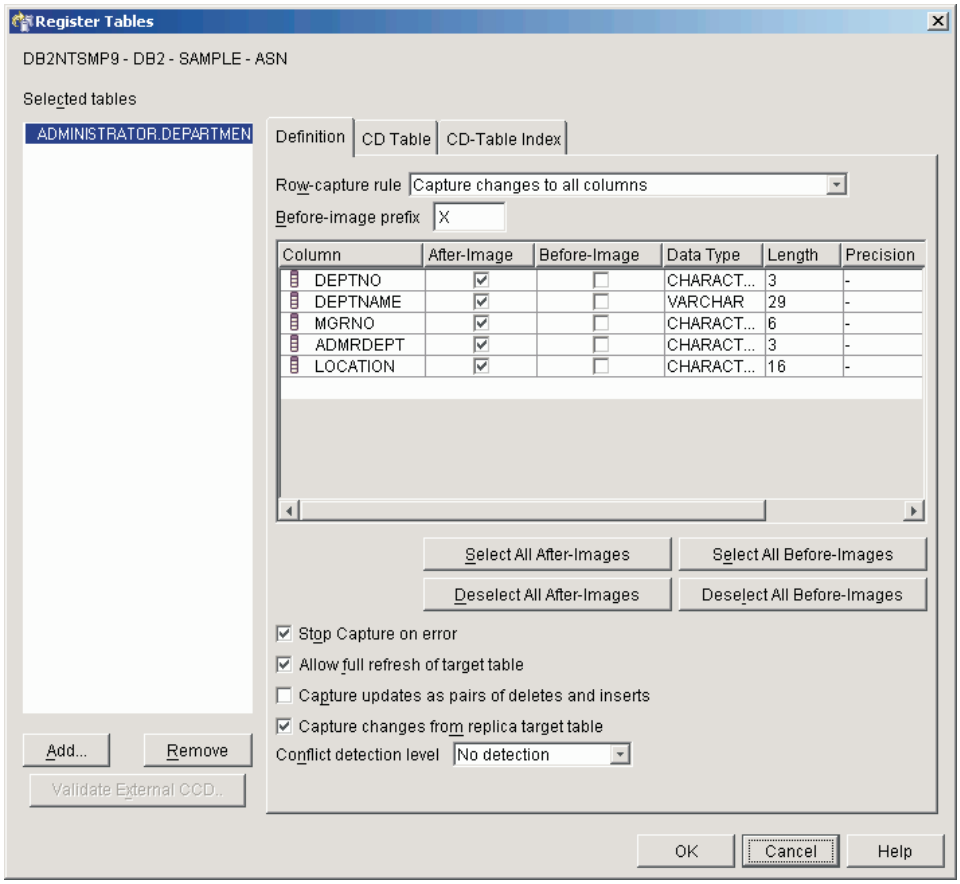


Figure 8. The Register Tables window

9. In the Register Tables window, click the **CD Table** notebook tab. Specify the following information for the CD table space:
   - In the **Specification for table space** area, click the **Container name** field to specify the container name for the TSCDDEPARTMENT table space.
   - In the **Specification for table space** area, change the **Size** field to 1.
   - In the **Specification for table space** area, change the **Unit** field to MB.
   - Specify the other information for this new table space; for example, set the buffer pool to IBMDEFAULTBP.

   After you have entered the table-space information, click **OK**.
10. Click **Close** on the Message Dialog window. This window shows the result of generating the SQL script that will register the source table. If there were any errors, they would be displayed in this window.
11. Click **OK** on the Run Now or Save SQL window to run the SQL script immediately.
12. You should see a message in the DB2 Message window that the script ran successfully. Click **Close**.
13. The contents pane for the SAMPLE database folder should now show the DEPARTMENT table as a registered table. See Figure 9 on page 249 for an example of the contents pane for the SAMPLE database folder with the DEPARTMENT table as a registered table.

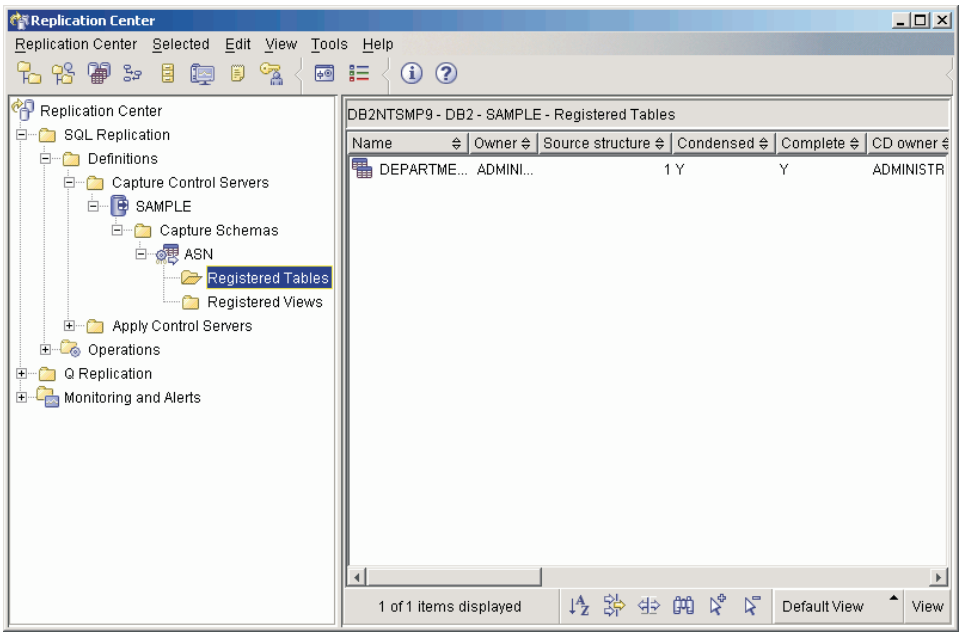


Figure 9. The DEPARTMENT table is listed as a registered table for the SAMPLE database

The DEPARTMENT table is now defined as a replication source. When you ran the SQL script, the Replication Center created the CD table and CD-table index for this replication source, and it updated the Capture control tables.

Step 4: Create replication control tables for the Apply program

The Apply program reads the replication control tables for current subscription-set information and stores its status in these tables. Any database that will act as an Apply control server must contain the Apply control tables.

To create Apply control tables:

1. Expand the **SQL Replication** folder.
2. Expand the **Definitions** folder.
3. Right-click the **Apply Control Servers** folder and select **Create Apply Control Tables → Quick**. Alternatively, you could customize the Apply control tables by selecting **Create Apply Control Tables → Custom**.
4. In the Select a Server window, select the COPYDB database. This database will be your Apply control server. Click **OK**.
5. In the Create Control Tables - Quick - Server Information window, select **Apply captured changes to target tables**. Then click **Next**.
6. In the Create Control Tables - Quick - Replication Details window, click **Next**. You do not need to change any of the information in this window.
7. In the Create Control Tables - Quick - Table Spaces window, enter the table space specification for the TSASNAA table space. For example, set the buffer pool to IBMDEFAULTBP. Click **OK**.
8. Click **Close** on the Message Dialog window. This window shows the result of generating the SQL script that will create the Apply control tables. If there were any errors, they would be displayed in this window.
9. Enter a valid user ID and password in the Run Now or Save SQL window and click **OK** to run the SQL script immediately.

10. You should see a message in the DB2 Message window that the script ran successfully. Click **Close**.
11. Expand the **Apply Control Servers** folder. The COPYDB database should be displayed under the folder.

Step 5: Create a subscription set and a subscription-set member

After you register the source table, you need to create a subscription set. A subscription set defines a relationship between the replication source database (SAMPLE in this scenario) and a target database (COPYDB in this scenario). A subscription-set member defines a relationship between the replication source table (DEPARTMENT in this scenario) and one or more target tables (this scenario has only one, it will be called DEPTCOPY).

To create a subscription set and a subscription-set member:

1. Expand the **SQL Replication** folder.
2. Expand the **Definitions** folder.
3. Expand the **Apply Control Servers** folder.
4. Expand the COPYDB database.
5. Right-click the **Subscription Sets** folder and select **Create**.

   You can also create a subscription set by selecting the **Registered Tables** folder of the SAMPLE database, right-clicking the DEPARTMENT table in the contents pane, and selecting **Create Subscription Set**.
6. In the Set Information page of the Create Subscription Set window, enter the following information:
   a. In the **Set name** field, enter `DEPTSUB`. This string identifies the subscription set and must be unique for a particular Apply qualifier.
   b. In the **Apply qualifier** field, enter `DEPTQUAL`. This string identifies the replication definitions that are unique to the instance of the Apply program that will run this subscription set.

      Tip: The Apply qualifier is case-sensitive. If you want the Apply qualifier to be in lowercase characters, you must delimit it when you type it; for example, `"deptqual"`. If you simply type `deptqual`, the Replication Center converts the value to uppercase characters by default.
   c. Click the browse button for the **Capture control server alias** field. In the Select a Capture Control Server window, select the SAMPLE database and click **OK**.
   d. Click the browse button for the **Target server alias** field. In the Select a Target Server window, select the COPYDB database and click **OK**. The COPYDB database is both the target server and the Apply control server.
   e. Select the **Activate the subscription set** check box.

   You do not need to change the settings of the other fields in the Set Information page. The Create Subscription Set window should look similar to the window shown in Figure 10 on page 251.

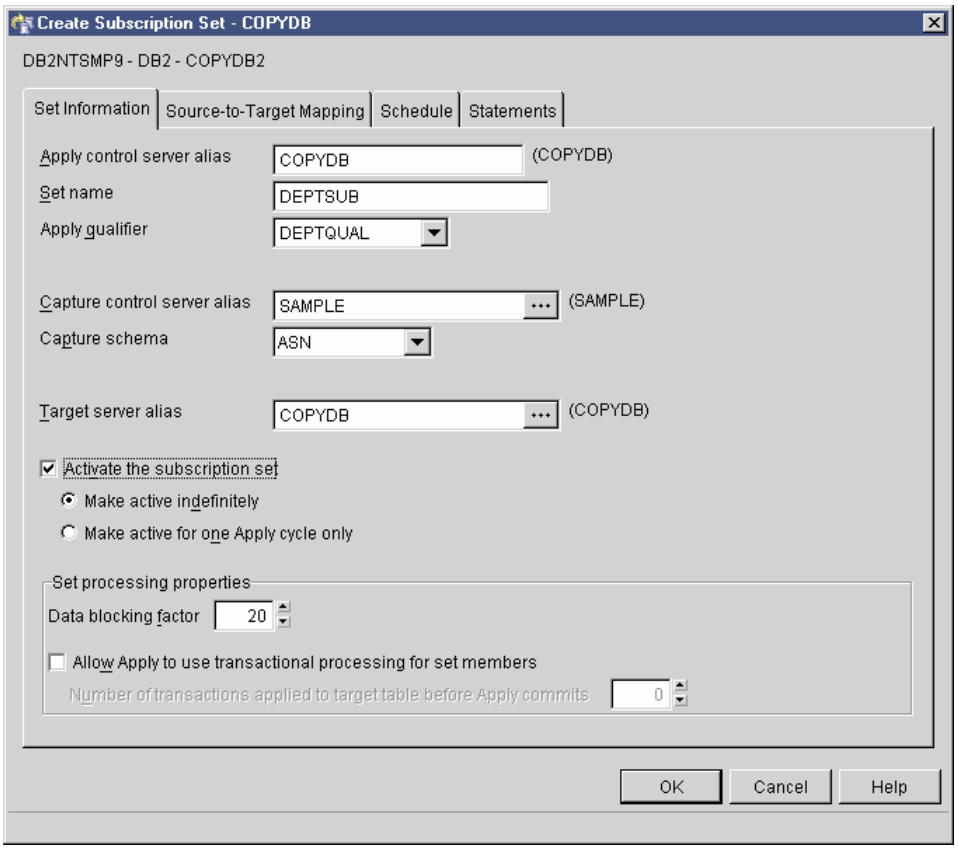


Figure 10. The Create Subscription Set window

7. In the Source-to-Target Mapping page of the Create Subscription Set window, enter the following information:
   a. Click **Add** to add a registered source to the subscription-set member.
   b. In the Add Registered Sources window, click **Retrieve All** to display all registered sources in the SAMPLE database.
   c. In the Add Registered Sources window, select the DEPARTMENT table and click **OK**.
   d. In the Source-to-Target Mapping page of the Create Subscription Set window, change the name of the target table from TGDEPARTMENT to TGDEPTCOPY: select TGDEPARTMENT in the **Target name** column of the Subscription-set members table, and type `TGDEPTCOPY` over the default name.

      Do not change the target type because you want to create a user copy target table.
   e. Click **Change** to open the Member Properties window. From this window, you can define the properties for the subscription-set member.

      Because for this scenario you want to replicate all columns and create the same columns in the target table as in the source table, you do not need to make any changes to the Column Selection or Column Mapping pages of the Member Properties window. By default, the target table contains all of the columns that you registered for the source.

      Tip for Federated: If your company's replication configuration maps a source table to an existing target table and at least one of the tables is from

a non-DB2 relational database, see the *Federated Systems Guide* for more information on how to map the source columns to columns in the existing target table.

An example of the Member Properties window is shown in Figure 11.

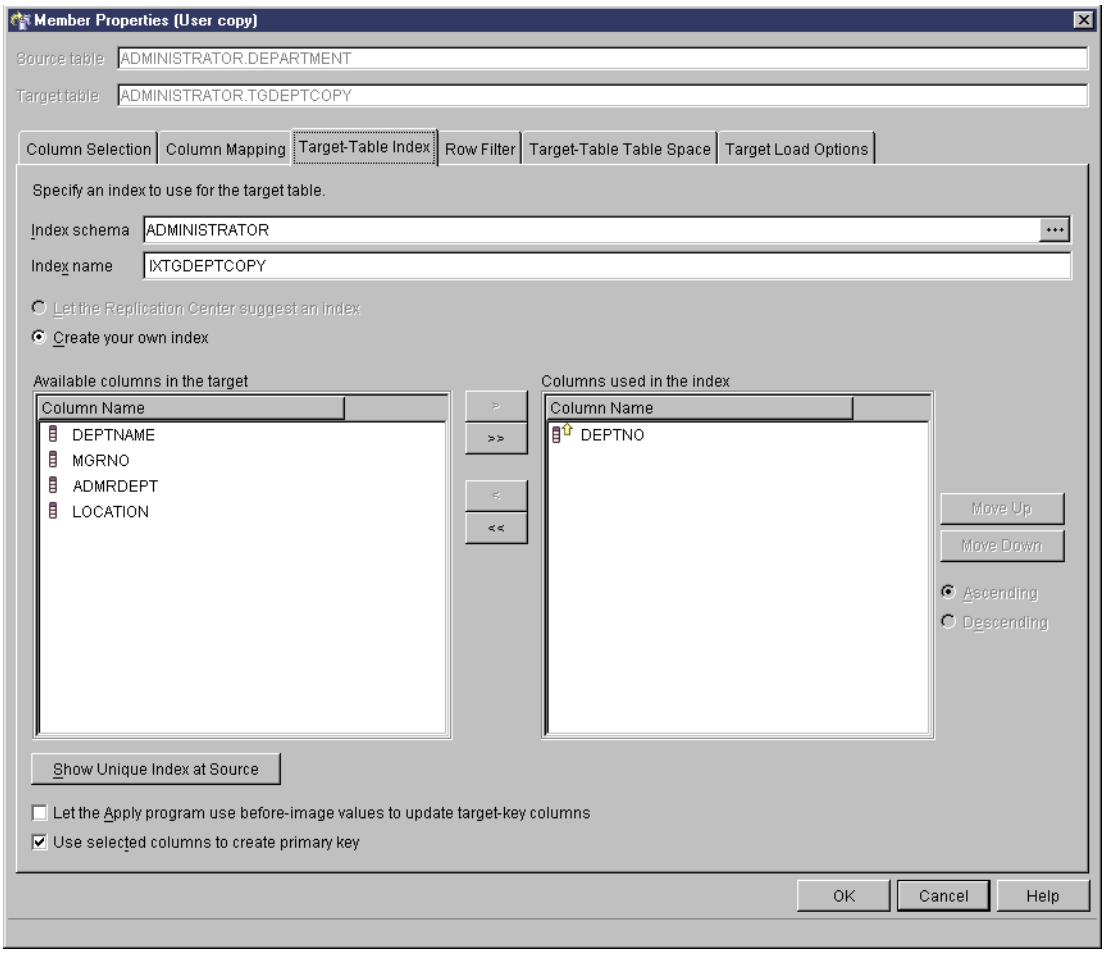


Figure 11. The Member Properties window

8. In the Target-Table Index page of the Member Properties window:
   a. Select the DEPTNO column from the **Available columns in the target** list.
   b. Click the move button (>) to move the DEPTNO column to the **Columns used in the index** list.
   c. Select **Use selected columns to create primary key** to use the DEPTNO column as the primary key for the target table.
9. In the Row Filter page of the Member Properties window, enter the following clause in the **WHERE statement** field:

   ```
   DEPTNO >='E00'
   ```

   This WHERE clause indicates that you want to replicate only those rows that meet certain criteria; in this case, that the department number is greater than or equal to “E00”. This WHERE clause will cause the target table to contain three rows, instead of all nine rows.
10. In the Target-Table Table Space page of the Member Properties window, specify the following information for the new TSTGDEPTCOPY table space:

- In the **Specification for table space** area, click the **Container name** field to specify the container name for the TSTGDEPTCOPY table space.
- In the **Specification for table space** area, change the **Size** field to 1.
- In the **Specification for table space** area, change the **Unit** field to MB.
- Specify the other information for this new table space; for example, set the buffer pool to IBMDEFAULTBP.

You can also specify the other information for this new table space; for example, set the buffer pool to IBMDEFAULTBP.

11. Click **OK** to close the Member Properties window. For this scenario, you do not need to do anything on the **Target load options** tab.
12. In the Schedule page of the Create Subscription Set window, change the number of minutes to 1 so that the Apply program will process this subscription set every minute. Use the spin button on the **Minutes** field in the **Frequency of replication** area to select one-minute intervals (or type 1 in the field).

    Keep the default values for **Start date**, **Start time**, **Time-based**, and **Use relative timing**.
13. In the Statements page of the Create Subscription Set window, click **Add** to open the Add SQL Statement or Procedure Call window. Use this window to define the SQL statements that will be processed when the subscription set is run. In the Add SQL Statement or Procedure Call window, enter the following information:
    a. In the **SQL Statement** field, enter:

    ```
    DELETE FROM ASN.IBMSNAP_APPLYTRAIL WHERE LASTRUN
          < (CURRENT TIMESTAMP - 7 DAYS)
    ```

    This statement deletes any records in the Apply trail table that are older than seven days.

    The Apply program will execute the SQL statement that you added at the target server after the subscription set is processed. The SQL statement must run at the target server because the Apply control server and target server are co-located and the Apply trail table is in the Apply control server.

    Tip: The Apply program runs SQL statements or procedures that you add to a subscription set during every subscription cycle. This example is inefficient because the Apply program will execute this statement every minute, even though the statement will only delete data from the APPLYTRAIL table at most once every 24 hours.

    b. In the **SQLSTATE** field, enter `02000` and click **Add**. This SQL state indicates that ″row not found″ error is acceptable and that the Apply program should ignore these errors.

    Tip: You can define up to ten SQL states that you want the Apply program to ignore for this subscription set.

    c. Click OK to close the Add SQL Statement or Procedure Call window.
14. Click **OK** to close the Create Subscription Set window.
15. Click **Close** on the Message Dialog window. This window shows the result of generating the SQL script that will update the Apply control tables and create the target table. If there were any errors, they would be displayed in this window.

16. Click **OK** on the Run Now or Save SQL window to run the SQL script immediately.

    You could save the SQL script to a file for future use and also run it immediately:

    a. Select **Save to file**.
    b. Fill in the information in the **Save specifications** area, such as the file name.
    c. Click Apply to save the file. If the script has multiple parts and you did not select the **Save multiple scripts in one file** check box, then each part is saved to a separate file using the name that you specify plus a number. The Run Now or Save SQL window remains open.
    d. Select **Run now**.
    e. Click **OK** to run the script and close the Run Now or Save SQL window.

    You can also save the SQL script to a file to run it later, or you can save the SQL script and run it

17. You should see a message in the DB2 Message window that the script ran successfully at both the SAMPLE and COPYDB servers. Click **Close**.
18. Expand the **Apply Control Servers** folder and the COPYDB database, then click the **Subscription Sets** folder. The contents pane for the **Subscription Sets** folder should now show the DEPTSUB subscription set. See Figure 12 for an example of the contents pane for the **Subscription Sets** folder with the DEPTSUB subscription set.

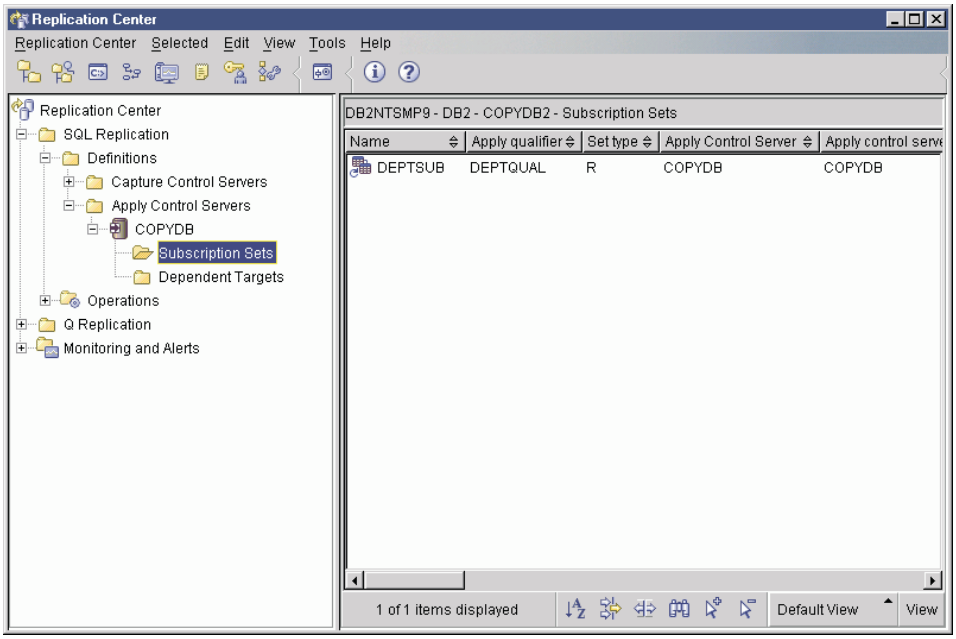


Figure 12. The DEPTSUB subscription set is listed for the COPYDB database

Step 6: Create an Apply password file

Because the Apply program needs to connect to the Capture control server, the Apply control server, and the target server, you must create a password file for user authentication. Because the contents of the password file are encrypted, only the Apply program can read the file, although you can modify the file.

To create a password file:

1. Open a Windows command prompt window and change to the C:\sqllib\bin directory.
2. Enter the following command to create a default password file:

   ```
   asnpwd init using "path"
   ```

   where *path* is the fully specified directory path and file name that you want to use when you create the password file. You should see message ASN1981I that confirms that the command completed successfully.

   For example, if you want to store the password file in the c:\sqllib\repl directory and name the file asnpwd.aut, enter the following command:

   ```
   asnpwd init using "c:\sqllib\repl\asnpwd.aut"
   ```

   Tip: Create the password file in the directory in which you will start the Apply program. When you start the Apply program, you specify the file name for the password file (using the PWDFILE keyword) and the value for the directory in which the Apply program will store its log and work files (using the APPLY_PATH keyword). One of the Apply program's work files is the password file.
3. Enter the following command to add the user ID and password information for each database to which the Apply program must connect:

   ```
   asnpwd add alias SAMPLE id userid password password using "path"
   ```

   where *userid* is a valid DB2 user ID with sufficient authority to update the Capture and Apply control tables. You should see message ASN1981I that confirms that the command completed successfully.

Step 7: Replicate the scenario data

After you register the replication source and create the subscription set, start the Capture and Apply programs to perform the initial full refresh for the target table and begin change-capture replication.

To start the Capture program:

1. Expand the **SQL Replication** folder.
2. Expand the **Operations** folder.
3. Select the **Capture Control Servers** folder. The SAMPLE database should be displayed in the contents pane for Capture control servers.
4. Right-click the SAMPLE database and select **Start Capture**.
5. In the Start Capture window, select ASN in the **Capture schema** field.
6. Click **OK** on the Start Capture window.
7. Click **OK** on the Run Now or Save Command window to run the command immediately.
8. You should see a message in the DB2 Message window that the command ran successfully. Click **Close**. The Capture program is now running, but will not begin capturing changes for registered tables until the Apply program completes a full refresh for all registered tables.

To start the Apply program:

1. Expand the **SQL Replication** folder.
2. Expand the **Operations** folder.
3. Expand the **Apply Control Servers** folder.
4. Expand the COPYDB database.

5. Select the **Apply Qualifiers** folder. The DEPTQUAL Apply qualifier for subscription set DEPTSUB should be displayed in the contents pane for Apply qualifiers.
6. Right-click the DEPTQUAL Apply qualifier and select **Start Apply**.
7. In the Start Apply window, click the browse button for the **Where Apply is running** field to select the system or IP address on which to run the Apply program.
8. Click **OK** on the Start Apply window.
9. If necessary, type a valid user ID and password for the system on which you will run the Apply program in the Run Now or Save Command window.
10. Click **OK** on the Run Now or Save Command window to run the command immediately.
11. You should see a message in the DB2 Message window that the command ran successfully. Click **Close**. The Apply program is now running.

If you view the TGDEPTCOPY target table after one replication cycle, you should see results that match the data shown in Table 23. You can use any of the following methods to view the contents of the table:

- Use the Replication Center:
  1. Expand the **SQL Replication** folder.
  2. Expand the **Definitions** folder.
  3. Expand the **Apply Control Servers** folder.
  4. Expand the COPYDB database.
  5. Right-click the **Dependent Targets** folder and select **View Selected Contents**.
- Use the DB2 Control Center:
  1. Expand the databases folder for your DB2 instance.
  2. Expand the COPYDB database.
  3. Select the **Tables** folder.
  4. Right-click the TGDEPTCOPY table in the contents pane and select **Sample Contents**.
- Use the DB2 Command Center or a DB2 command window to issue the following SQL statement:

  ```
  SELECT * FROM schema.TGDEPTCOPY
  ```

Table 23. The TGDEPTCOPY table

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
E01	SUPPORT SERVICES	000050	A00	-
E11	OPERATIONS	000090	E01	-
E21	SOFTWARE SUPPORT	000100	E01	-

Operating in a replication environment

After the replication environment is up and running, changes that you make to the replication source table will be replicated to the target table. You can view status for both the Capture and Apply programs to understand your replication latency and other information about your replication environment. And although the Capture and Apply programs can run continuously, there are times when you will want to stop them (for example, to run utilities that use the table spaces that contain the control tables).

Step 1: Update the source table

Assume that two new departments were created at the Spiffy Computer Service: a technical writing department and a public relations department. Your target table will include both of these departments.

To update the source table:

1. Select **Start → Programs → IBM DB2 → Command Window** to open a DB2 command window.
2. Connect to the source server:

   ```
   DB2 CONNECT TO SAMPLE
   ```

3. Add two new rows, one for each department, by typing each of the following commands and pressing Enter after each one:

   ```
   DB2 INSERT INTO DEPARTMENT
       VALUES ('F01','TECHNICAL WRITING','000110','F01',NULL)
   DB2 INSERT INTO DEPARTMENT
       VALUES ('G01','PUBLIC RELATIONS','000120','G01',NULL)
   DB2 COMMIT
   ```

4. Connect to the target server:

   ```
   DB2 CONNECT TO COPYDB
   ```

5. Wait at least one minute, then verify that the new rows are replicated to the target database by typing the following command and pressing Enter:

   ```
   DB2 SELECT * FROM TGDEPTCOPY
   ```

   You must wait for one minute because the subscription set is eligible for replication every minute. If there were a large amount of data, you might want to wait a bit longer for the Apply to apply that data to the target table.

Table 24 shows the results of the replication, with two new rows appended to the table.

Table 24. The TGDEPTCOPY table after changes are replicated

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
E01	SUPPORT SERVICES	000050	A00	-
E11	OPERATIONS	000090	E01	-
E21	SOFTWARE SUPPORT	000100	E01	-
F01	TECHNICAL WRITING	000110	F01	-
G01	PUBLIC RELATIONS	000120	G01	-

Step 2: View status for the Capture program

Use the Replication Center to view the following status information for the Capture program:

- Error messages issued by the Capture program
- An analysis of the Capture program's throughput
- A summary of the Capture program's current latency
- The current operational status for the Capture program

Each of these kinds of status give you a snapshot view of how the Capture program is running.

To query the status of the Capture program:

1. Expand the **SQL Replication** folder.
2. Expand the **Operations** folder.
3. Select the **Capture Control Servers** folder.
4. Right-click the SAMPLE database in the contents pane and select **Check Status**.
5. Click the yellow arrow icon to view current information.

To view an analysis of the Capture program's throughput:

1. Expand the **SQL Replication** folder.
2. Expand the **Operations** folder.
3. Select the **Capture Control Servers** folder.
4. Right-click the SAMPLE database in the contents pane and select **Show Capture Throughput Analysis**.
5. In the Capture Throughput Analysis window, you can view the following information:
   - The number of rows inserted to the CD tables from the DB2 log or skipped for various reasons, such as the setting of the CHGONLY keyword
   - The number of rows pruned from the CD tables
   - The number of transactions committed by the Capture program
   - The memory usage of the Capture program within a specific time interval
6. Click **Retrieve** to view current information.

   Note: After the first attempt to retrieve any data, the label of the button changes to **Refresh**.

To view a summary of the Capture program's current latency:

1. Expand the **SQL Replication** folder.
2. Expand the **Operations** folder.
3. Select the **Capture Control Servers** folder.
4. Right-click the SAMPLE database in the contents pane and select **Show Capture Latency**.
5. In the Capture Latency window, you can view the average, minimum, and maximum latency for the Capture program within a specific time interval.
6. Click **Retrieve** to view current information.

   Note: After the first attempt to retrieve any data, the label of the button changes to **Refresh**.

Step 3: View status for the Apply program

Use the Replication Center to view the following status information for the Apply program:

- A summary of subscription-set information, including successful and failed subscription sets
- A summary of the performance of the Apply program
- A summary of the end-to-end replication latency
- The current operational status for the Apply program

Each of these kinds of status give you a snapshot view of how the Apply program is running.

To query the status of the Apply program:

1. Expand the **SQL Replication** folder.
2. Expand the **Operations** folder.
3. Select the **Apply Control Servers** folder.
4. Expand the COPYDB database.
5. Select the **Apply Qualifiers** folder.
6. Right-click the DEPTQUAL Apply qualifier in the contents pane and select **Check Status**.
7. If the **Where Apply is running** field is empty, select the system or IP address on which the Apply program is running.
8. Click the yellow arrow icon to view current information.

To view a summary of Apply program performance:

1. Expand the **SQL Replication** folder.
2. Expand the **Operations** folder.
3. Expand the **Apply Control Servers** folder.
4. Expand the COPYDB database.
5. Select the **Apply Qualifiers** folder.
6. Right-click the DEPTQUAL Apply qualifier in the contents pane and select **Show Apply Throughput Analysis**.
7. In the Show Apply Throughput Analysis window, you can view the following information:
   - The number of rows fetched by the Apply program from the CD tables
   - The elapsed time for each subscription set
8. Click **Retrieve** to view current information.

   Note: After the first attempt to retrieve any data, the label of the button changes to **Refresh**.

To view a summary of the end-to-end replication latency:

1. Expand the **SQL Replication** folder.
2. Expand the **Operations** folder.
3. Expand the **Apply Control Servers** folder.
4. Expand the COPYDB database.
5. Select the **Apply Qualifiers** folder.
6. Right-click the DEPTQUAL Apply qualifier in the contents pane and select **Show End-to-End Latency**.
7. In the Show End-to-End Latency window, you can view the average latency for each subscription set within a specific time interval.
8. Click **Retrieve** to view current information.

   Note: After the first attempt to retrieve any data, the label of the button changes to **Refresh**.

Step 4: Stop the Capture and Apply programs

An important part of maintaining your replication environment is regular database maintenance. Sometimes that maintenance will require you to stop the Capture and Apply programs. For example, you must stop the Capture and Apply programs before you run utilities that directly use the table spaces that are used by these programs.

To stop the Capture program:

1. Expand the **SQL Replication** folder.
2. Expand the **Operations** folder.
3. Select the **Capture Control Servers** folder.
4. Right-click the SAMPLE database in the contents pane and select **Stop Capture**.
5. Click **OK** on the Stop Capture window.
6. Click **OK** on the Run Now or Save Command window to run the command immediately.
7. You should see a message in the DB2 Message window that the command ran successfully. Click **Close**. The Capture program is now stopped.

To stop the Apply program:

1. Expand the **SQL Replication** folder.
2. Expand the **Operations** folder.
3. Expand the **Apply Control Servers** folder.
4. Expand the COPYDB database.
5. Select the **Apply Qualifiers** folder.
6. Right-click the DEPTQUAL Apply qualifier in the contents pane and select **Stop Apply**.
7. In the Stop Apply window, click **OK**. The Apply program is now stopped.

You can run DB2 utilities on your database now that you have stopped the Capture and Apply programs. Running utilities is beyond the scope of this scenario.

Monitoring replication

After the replication environment is up and running, there will be times when you want to understand how well the Capture and Apply programs are running. You might also want to set up automatic notifications in the event of certain kinds of replication errors.

You can use the Replication Center to query the status of the Capture and Apply programs and to view certain statistics that can tell you how well either program is running. You can also set up the Replication Alert Monitor to notify you when either the Capture or Apply programs encounter certain kinds of replication errors.

Step 1: Create replication control tables for the Monitor program

The Replication Alert Monitor program reads the replication monitor control tables for current monitor information and stores its status in these tables. Any database that will act as a Monitor server must have the Monitor control tables.

To create Monitor control tables:

1. Expand the **Monitoring and Alerts** folder.
2. Right-click the **Monitor Control Servers** folder and select **Create Monitor Control Tables**.
3. In the Select a Server window, select the COPYDB database. This database will be your Monitor control server.
4. In the Create Monitor Control Tables window, select the IBMSNAP_CONTACTS control table and fill in the information for the

RELPMONTS1 table space properties. Click the browse button next to the container name to customize the location for this table space. You can also specify other information for the table space, for example, set the buffer pool to IBMDEFAULTBP. By default, all of the Monitor control tables except IBMSNAP_ALERTS and IBMSNAP_MONTRACE share the same table space as the IBMSNAP_CONTACTS table. The Create Monitor Control Tables window should look similar to the window shown in Figure 13.

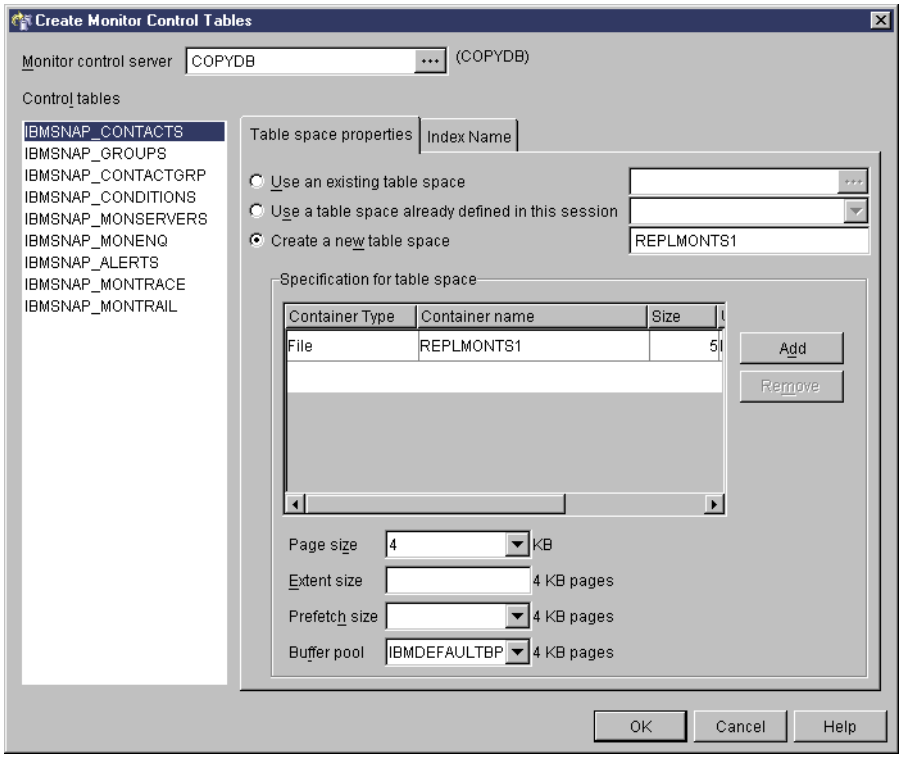


Figure 13. The Create Monitor Control Tables window

5. In the Create Monitor Control Tables window, select the IBMSNAP_ALERTS control table and fill in the information for the RELPMONTS2 table space properties.
6. In the Create Monitor Control Tables window, select the IBMSNAP_MONTRACE control table and fill in the information for the RELPMONTS3 table space properties.
7. Click **OK** on the Create Monitor Control Tables window to accept the default values for the other control table information, including index names.
8. Click **Close** on the Message Dialog window. This window shows the result of generating the SQL script to create the Monitor control tables. If there were any errors, they would be displayed in this window.
9. Click **OK** on the Run Now or Save SQL window to run the SQL script immediately.
10. You should see a message in the DB2 Message window that the script ran successfully. Click **Close**.
11. Expand the **Monitor Control Servers** folder. The COPYDB database should be displayed under the folder.

Step 2: Create a contact for replication alerts

The Replication Alert Monitor program can alert you when it detects specific activity of the Capture and Apply programs. You can create individual contacts or groups of contacts if the Replication Alert Monitor should alert several people for a specific alert condition.

To create a contact:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand the COPYDB database.
4. Right-click the Contacts folder and select **Create Contact → Person**.
5. In the Create Contact window, enter your name and e-mail address. Click **OK** to close the window.
6. Click **Close** on the Messages and SQL Scripts window. This window shows the result of generating the SQL script to update the Monitor control tables. If there were any errors, they would be displayed in this window.
7. Click **OK** on the Run Now or Save SQL window to run the SQL script immediately.
8. You should see a message in the DB2 Message window that the script ran successfully. Click **Close**.
9. Click the **Contacts** folder. The contact that you defined should be displayed in the contents pane for Contacts.

Step 3: Select alert conditions for the Capture program

The Replication Alert Monitor program can monitor specific activity of the Capture program. You must select which activities you want to monitor. For each of these activities, you select an alert condition. When the Capture program encounters the condition, the Replication Alert Monitor sends an alert to those contacts that you define for the alert condition.

To create Monitor definitions for the Capture program:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand the COPYDB database.
4. Right-click the **Monitors** folder and select **Create**.
5. In the Create Monitor Wizard, enter the following information:
   a. In the **Monitor qualifier** field, type `MON1`.
   b. Under **SQL Replication**, select the **Capture programs** check box.
   c. Click **Next**.
   d. Click **Add** on the Select alert conditions for Capture programs page.
   e. Click the browse button for the **Capture control server** field to select the Capture control server that you want to monitor. In the Select a Capture Control Server window, select the SAMPLE database and click **OK**.
   f. In the table, select **CAPTURE_ERRORS** check box.
   g. Place your cursor in the **Contact** field that is in the same row as CAPTURE_ERRORS and click the browse button.
   h. In the Select Contact or Contact Group window, select the contact that you created in "Step 2: Create a contact for replication alerts" and click **OK** to close the window.

i. Click **OK** to close the Select Alert Conditions for Capture Schemas window.
j. Click **Finish**.

6. Click **Close** on the Messages and SQL Scripts window. This window shows the result of generating the SQL script to update the Monitor control tables. If there were any errors, they would be displayed in this window.
7. Click **OK** on the Run Now or Save SQL window to run the SQL script immediately.
8. You should see a message in the DB2 Message window that the script ran successfully. Click **Close**.
9. Expand the COPYDB database, expand the **Monitors** folder, and select the MON1 folder. The alert conditions that you defined should be displayed in the contents pane for Monitor qualifiers.

Step 4: Select alert conditions for the Apply program

The Replication Alert Monitor program can monitor specific activity of the Apply program. You must select which activities you want to monitor. For each of these activities, you select an alert condition. When the Apply program encounters the condition, the Replication Alert Monitor sends an alert to those contacts that you define for the alert condition.

To create Monitor definitions for the Apply program:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand the COPYDB database.
4. Right-click the **Monitors** folder and select **Create**.
5. In the Create Monitor Wizard, enter the following information:
   a. In the **Monitor qualifier** field, type `MON1` if you did not create alert conditions for the Capture program.
   b. Under **SQL Replication**, select the **Apply program by Apply qualifier** check box.
   c. Click **Next**.
   d. Click **Add** on the Select alert conditions for Apply programs by Apply qualifiers, including all subscription sets that they process page.
   e. Click the browse button for the **Apply control server** field to select the Apply control server that you want to monitor. In the Select an Apply Control Server window, select the COPYDB database and click **OK**.
   f. In the table, select **APPLY_FULLREFRESH** check box.
   g. Place your cursor in the **Contact** field that is in the same row as APPLY_FULLREFRESH and click the browse button.
   h. In the Select Contact or Contact Group window, select the contact that you created in "Step 2: Create a contact for replication alerts" on page 262 and click **OK** to close the window.
   i. Click **OK** in the Select Alert Conditions for Apply Qualifiers or Subscription Sets window.
   j. Click **Finish**.
6. Click **Close** on the Messages and SQL Scripts window. This window shows the result of generating the SQL script to update the Monitor control tables. If there were any errors, they would be displayed in this window.
7. Click **OK** on the Run Now or Save SQL window to run the SQL script immediately.

8. You should see a message in the DB2 Message window that the script ran successfully. Click **Close**.
9. Expand the COPYDB database, expand the **Monitors** folder, and select the MON1 folder. The alert conditions that you defined should be displayed in the contents pane for Monitor qualifiers. See Figure 14 for an example of the contents pane for the **Monitor Qualifiers** folder with the MON1 monitor qualifier.

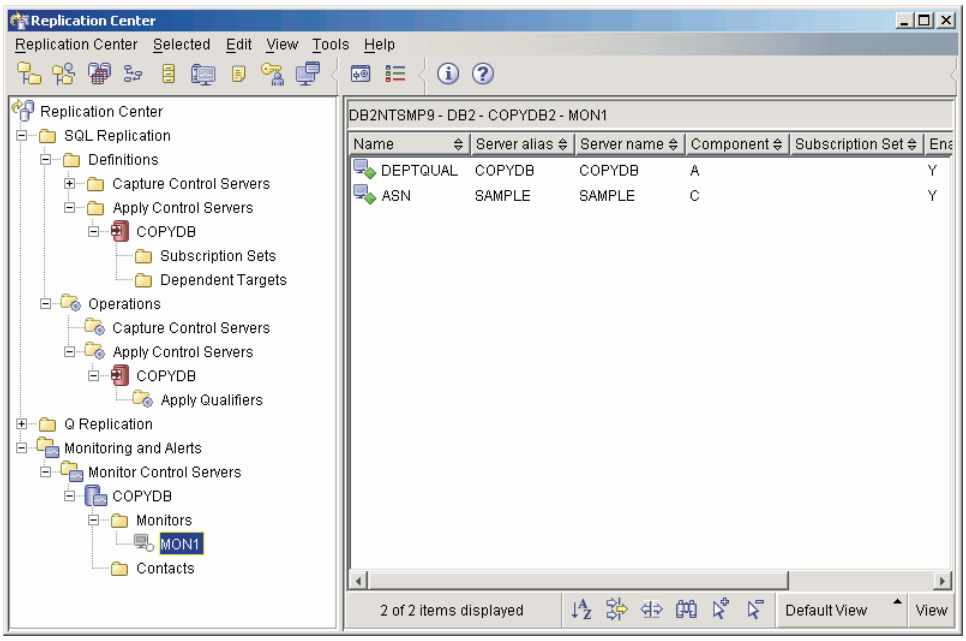


Figure 14. The MON1 monitor qualifier is listed for the COPYDB database

Step 5: Start the Replication Alert Monitor for a monitor qualifier

After you select alert conditions for the Capture or Apply programs, you can start the Replication Alert Monitor program to monitor the activity of the Capture and Apply programs for the specific conditions associated with a monitor qualifier. When the Capture or Apply program encounters one of the specified conditions, the Replication Alert Monitor sends an alert to those contacts that you defined for the alert condition.

To start the Replication Alert Monitor:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand the COPYDB database.
4. Expand the **Monitors** folder.
5. Right-click the MON1 monitor qualifier and select **Start Monitor**.
6. In the Start Monitor window, enter the following information:
   a. Select the MONITOR_PATH keyword. Enter a value for the directory in which the Replication Alert Monitor will store its log and work files.

      Tip: Set the value for the MONITOR_PATH keyword to the value that you set for the APPLY_PATH keyword so that both the Replication Alert Monitor and the Apply program can use the same password file.

b. Select the EMAIL_SERVER keyword. Enter your e-mail server name.
c. Select the MONITOR_ERRORS keyword. If you want the Replication Alert Monitor to notify you if it encounters the conditions that you specified, enter your e-mail address. If you want the Replication Alert Monitor to notify someone else, click the browse button to open the Select Contact or Contact Group window to select a contact person or contact group.
d. Click **OK** to close the Start Monitor window.

7. Click **OK** on the Run Now or Save Command window to run the command immediately.
8. You should see a message in the DB2 Message window that the command started successfully. Click **Close**.

To display the alerts that the Replication Alert Monitor monitored:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand the COPYDB database.
4. Expand the **Monitors** folder.
5. Select the MON1 monitor qualifier.
6. In the contents pane for the monitor qualifier, right-click one of the alert conditions and select **Show Alerts**.
7. In the Show Alerts window, specify a time range and click **Retrieve**.

   Note: After the first attempt to retrieve any data, the label of the button changes to **Refresh**.

To stop the Replication Alert Monitor:

1. Expand the **Monitoring and Alerts** folder.
2. Expand the **Monitor Control Servers** folder.
3. Expand the COPYDB database.
4. Expand the **Monitors** folder.
5. Right-click the MON1 monitor qualifier and select **Stop Monitor**.
6. In the Stop Monitor window, click **OK**.
7. In the Run Now or Save Command window, type the directory in which you started the Replication Alert Monitor program in the **Directory** field, or use the browse button to select the path. Click **OK**.

   Tip: The **Directory** field is not enabled until after you specify the user id and password.
8. Click **OK** on the Run Now or Save Command window to run the command immediately.
9. You should see a message in the DB2 Message window that the command ran successfully. Click **Close**.

Part 3. Replication reference

This part of the book contains the following chapters:

Chapter 17, "Naming rules for SQL replication objects," on page 269 describes how to specify valid names for replication objects.

Chapter 18, "System commands for SQL replication (Linux, UNIX, Windows, z/OS)," on page 271 describes commands that experienced DB2 replication users can use instead of the Replication Center for operating replication on Linux, UNIX, Windows, and z/OS operating systems.

Chapter 19, "System commands for SQL replication (OS/400)," on page 319 describes the commands that you can use if you want to set up, administer, and maintain replication locally on the OS/400 operating system.

Chapter 20, "Operating the SQL replication programs (z/OS)," on page 405 describes how to start and operate the replication programs using JCL or system-started tasks on z/OS.

Chapter 21, "Using the Windows Service Control Manager to issue system commands for SQL replication (Windows)," on page 409 describes how to start replication programs as services on Windows operating systems.

Chapter 22, "Scheduling SQL replication programs on various operating systems," on page 413 describes how to schedule replication programs on various operating systems.

Chapter 23, "How the SQL replication components communicate," on page 415 describes how the replication components use the control tables to communicate with each other.

Chapter 24, "Table structures for SQL replication," on page 421 describes the table structures for the replication tables that reside on the various replication servers.

Chapter 17. Naming rules for SQL replication objects

The following table lists the limits for names of replication objects.

Table 25. Name limits for replication objects

Object	Name limits
Source and target tables	**UNIX, Windows, z/OS**: Follow the naming rules for your database management system. **OS/400**: Names cannot include blanks, asterisks (*), question marks (?), single quotation marks (’), double quotation marks (″), or a slash (/).
Source and target columns	Follow the naming rules for your database management system. (Note that all before-image columns have a one-character prefix added to them. To avoid ambiguous before-image column names, ensure that source column names are unique to 29 characters and that the before-image column names will not conflict with existing column names when the before-image character prefix is added to the column name.)
Subscription set	A subscription-set name can include any characters allowed by DB2 for varying-character (VARCHAR) columns. **Recommendation**: Follow the naming rules for DB2 table and column names. Because DB2 replication stores the subscription-set name in each replication control server, be sure that the name is compatible for all three servers’ code pages.
Capture schema	**UNIX, Windows**: The Capture schema can be a string of 30 or fewer characters[1]. **OS/390, z/OS**: The Capture schema can be a string of 18 or fewer characters; on DB2 UDB for z/OS Version 8 new-function mode subsystems it can be 128 characters[1]. **OS/400**: The Capture schema (CAPCTLLIB) can be a string of 10 or fewer alphanumeric characters[1].
Apply qualifier	**UNIX, Windows, z/OS**: The Apply qualifier can be a string of 18 or fewer characters[1]. **OS/400**: The Apply qualifier can be a string of 18 or fewer characters but, because Apply jobs can be only up to 10 characters long, the first 10 characters must be unique for a given Apply qualifier[1].
Monitor qualifier	**UNIX, Windows, z/OS**: The Monitor qualifier can be a string of 18 or fewer characters[1].

Table 25. Name limits for replication objects (continued)

Object	Name limits

Notes:

1. For Capture schemas, Apply qualifiers, and Monitor qualifiers, ensure that you use only the following valid characters in the names of these objects:
   - A through Z (uppercase letters)
   - a through z (lowercase letters)
   - Numerals (0 through 9)
   - The underscore character "_"

   Blanks are not allowed; neither are other special characters such as the colon ":" and the plus sign "+".

Replication system commands and the Replication Center, by default, convert all names that you provide to uppercase. Enclose a mixed-case character name in double quotation marks (or whatever character the target system is configured to use) to preserve the case and save the name exactly as you typed it. For example, if you type `myqual` or `MyQual` or `MYQUAL`, the name is saved as `MYQUAL`. If you type those same names and enclose them in double quotation marks, they are saved as `myqual` or `MyQual` or `MYQUAL`, respectively. Some operating systems don't recognize double quotation marks and you might have to use an escape character, typically a backslash (\).

On Windows operating systems, you *must* use a unique path to differentiate between names that are otherwise identical. For example, assume that you have three Apply qualifiers: `myqual` , `MyQual`, and `MYQUAL`. The three names use the same characters but different case. If these three qualifiers are in the same Apply path, they will cause name conflicts.

Important: When setting up Windows services for Capture, Apply, or the Replication Alert Monitor, you must use unique names for the Capture schema, Apply qualifier, and Monitor qualifier. You cannot use case to differentiate names.

Chapter 18. System commands for SQL replication (Linux, UNIX, Windows, z/OS)

This chapter describes the replication commands that run under one or more of the following operating systems:

- Linux
- UNIX
- Windows
- z/OS

All of these commands have a prefix of `asn` and are entered at an operating system command prompt or in a shell script. One of the commands, **asnanalyze**, also works with remote data residing on OS/400 operating systems.

This chapter contains a section for each command. Each section contains a brief description of the command, a syntax diagram, and a table of parameters with corresponding definitions. The end of each section has examples of command usage and cross-references to related information.

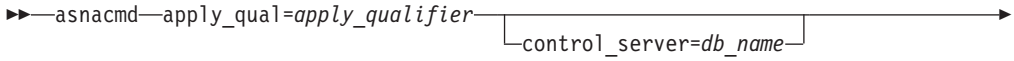
The commands include:

- "asnacmd: Operating Apply"
- "asnanalyze: Operating the Analyzer" on page 273
- "asnapply: Starting Apply" on page 276
- "asncap: Starting Capture" on page 282
- "asnccmd: Operating Capture" on page 288
- "asnmcmd: Working with a running Replication Alert Monitor" on page 292
- "asnmon: Starting a Replication Alert Monitor" on page 295
- "asnpwd: Creating and maintaining password files" on page 299
- "asnscrt: Creating a DB2 replication service to start the replication programs" on page 302
- "asnsdrop: Dropping DB2 replication services" on page 305
- "asnslist: Listing DB2 replication services" on page 306
- "asntdiff: Comparing data in source and target tables" on page 307
- "asntrc: Operating the replication trace facility" on page 309
- "asntrep: Repairing differences between source and target tables" on page 316

asnacmd: Operating Apply

Use the **asnacmd** command to operate the Apply program on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. Run this command at an operating system prompt or in a shell script.

Syntax



Parameters

Table 26 defines the invocation parameters.

Table 26. asnacmd invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
apply_qual=*apply_qualifier*	Specifies the Apply qualifier that the Apply program uses to identify the subscriptions sets to be served. You must specify an Apply qualifier. The value that you enter must match the value of the APPLY_QUAL column in the subscription set (IBMSNAP_SUBS_SET) table. The Apply qualifier name is case sensitive and can be a maximum of 18 characters.
control_server=*db_name*	Specifies the name of the Apply control server on which the subscription definitions and Apply control tables reside. **For Linux,UNIX and Windows:** If you do not specify an Apply control server, this parameter defaults to the value from the DB2DBDFT environment variable. **For z/OS:** The control server parameter is the name of the database server that connects to the control server.
status	Specify to receive messages that indicate the state of each thread (administration and worker) in Apply.
stop	Specify to stop the Apply program in an orderly way.

Examples for asnacmd

The following examples illustrate how to use the **asnacmd** command.

Example 1

To receive messages about the state of each Apply thread:

```
asnacmd apply_qual=AQ1 control_server=dbx status
```

Example 2

To stop the Apply program:

```
asnacmd apply_qual=AQ1 control_server=dbx stop
```

Related tasks:

- Chapter 20, "Operating the SQL replication programs (z/OS)," on page 405

Related reference:

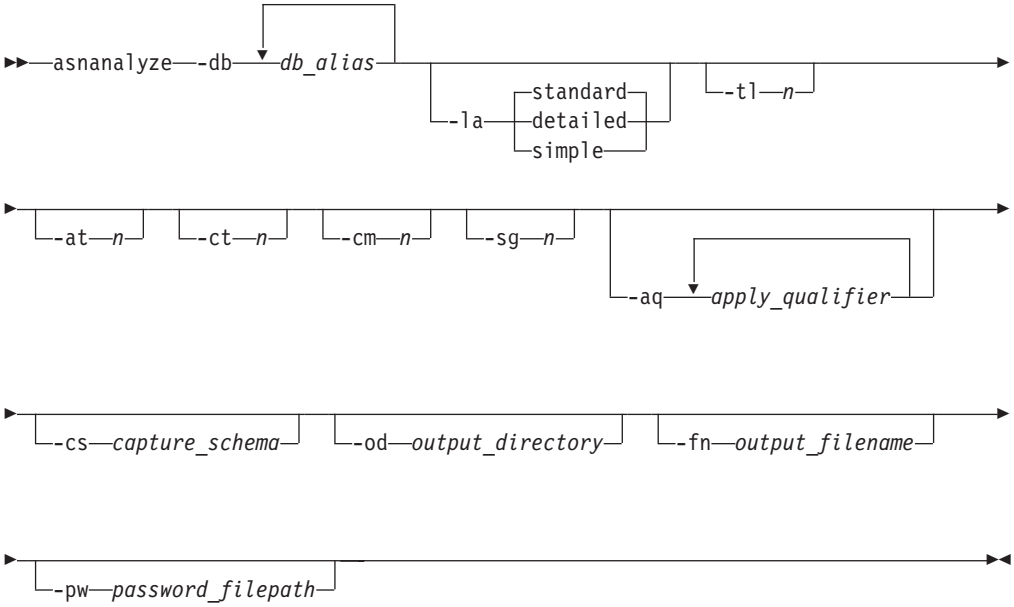
- "ENDDPRAPY: Stopping Apply (OS/400)" on page 361
- "STRDPRAPY: Starting Apply (OS/400)" on page 386

asnanalyze: Operating the Analyzer

Use the **asnanalyze** command to generate reports about the state of the replication control tables. This command analyzes replication control tables that reside on any operating system, including OS/400 operating systems; however, you must invoke the command from Linux, UNIX or Windows.

You must type a space between the **asnanalyze** command and the first parameter to invoke the command. If you issue the command without any parameters, you receive command help on the screen.

Syntax



Parameters

Table 27 defines the invocation parameters.

Table 27. asnanalyze invocation parameter definitions for Linux, UNIX and Windows operating systems

Parameter	Definition
-db *db_alias*	Specifies the Capture control server, target server, and Apply control server. You must provide at least one database alias. If there is more than one database alias, use blank spaces to separate the values.

Table 27. asnanalyze invocation parameter definitions for Linux, UNIX and Windows operating systems (continued)

Parameter	Definition
-la *level_of_analysis*	Specifies the level of analysis to be reported: **standard** (default) Generates a report that includes the contents of the control tables and status information from the Capture and Apply programs. **detailed** Generates the information in the standard report and: • Change-data (CD) and unit-of-work (UOW) table pruning information • DB2 for z/OS table space partitioning and compression information • Analysis of target indexes for subscription keys **simple** Generates the information in the standard report, but excludes the detail information from the subscription columns (IBMSNAP_SUBS_COLS) table.
-tl *n*	Specifies the date range (0 to 30 days) of entries to be retrieved from the Apply trail (IBMSNAP_APPLYTRAIL) table. The default is 3 days.
-at *n*	Specifies the date range (0 to 30 days) of entries to be retrieved from the Apply trace (IBMSNAP_APPLYTRACE) table. The default is 3 days.
-ct *n*	Specifies the date range (0 to 30 days) of entries to be retrieved from the Capture trace (IBMSNAP_CAPTRACE) table. The default is 3 days.
-cm *n*	Specifies the date range (0 to 30 days) of entries to be retrieved from the Capture monitor (IBMSNAP_CAPMON) table. The default is 3 days.
-sg *n*	Specifies the date range (0 to 30 days) of entries to be retrieved from the signal (IBMSNAP_SIGNAL) table. The default is 3 days.
-aq *apply_qualifier*	Specifies the Apply qualifier that identifies the specific subscription sets to be analyzed. You can specify more than one Apply qualifier. If there is more than one Apply qualifier, use blank spaces to separate the values. If no Apply qualifier is specified, all subscription sets for the specified database aliases are analyzed.
-cs *capture_schema*	Specifies the name of the Capture schema that you want to analyze. If you use this parameter, you can specify only one Capture schema.
-od *output_directory*	Specifies the directory in which you want to store the Analyzer report. The default is the current directory.

Table 27. asnanalyze invocation parameter definitions for Linux, UNIX and Windows operating systems (continued)

Parameter	Definition
-fn *output_filename*	Specifies the name of the file that will contain the Analyzer report output. Use the file naming conventions of the operating system that you are using to run the Analyzer. If the file name already exists, the file is overwritten. The default file name is asnanalyze.htm.
-pw *password_filepath*	Specifies the name and path of the password file. If you do not specify this parameter, the Analyzer checks the current directory for the asnpwd.aut file.

Examples for asnanalyze

The following examples illustrate how to use the **asnanalyze** command.

Example 1

To analyze the replication control tables on a database called proddb1:

```
asnanalyze -db proddb1
```

Example 2

To obtain a detailed level of analysis about the replication control tables on the proddb1 and proddb2 databases:

```
asnanalyze -db proddb1 proddb2 -la detailed
```

Example 3

To analyze the last two days of information from the IBMSNAP_APPLYTRAIL, IBMSNAP_APPLYTRACE, IBMSNAP_CAPTRACE, IBMSNAP_CAPMON, and IBMSNAP_SIGNAL tables on the proddb1 and proddb2 databases:

```
asnanalyze -db proddb1 proddb2 -tl 2 -at 2 -ct 2 -cm 2 -sg 2
```

Example 4

To obtain a simple level of analysis about the last two days of information from the IBMSNAP_APPLYTRAIL, IBMSNAP_APPLYTRACE, IBMSNAP_CAPTRACE, IBMSNAP_CAPMON, and IBMSNAP_SIGNAL tables on the proddb1 and proddb2 databases for only the qual1 and qual2 Apply qualifiers:

```
asnanalyze -db proddb1 proddb2 -la simple -tl 2 -at 2 -ct 2 -cm 2 -sg 2
  -aq qual1 qual2 -od c:\mydir -fn anzout -pw c:\SQLLIB
```

This command example writes the analyzer output to a file named anzout under the c:\mydir directory and uses the password information from the c:\SQLLIB directory.

Example 5

To analyze a specific Capture schema:

```
asnanalyze -db proddb1 proddb2 -cs BSN
```

Example 6

To display command help:

```
asnanalyze
```

Related reference:

• "ANZDPR: Operating the Analyzer (OS/400)" on page 352

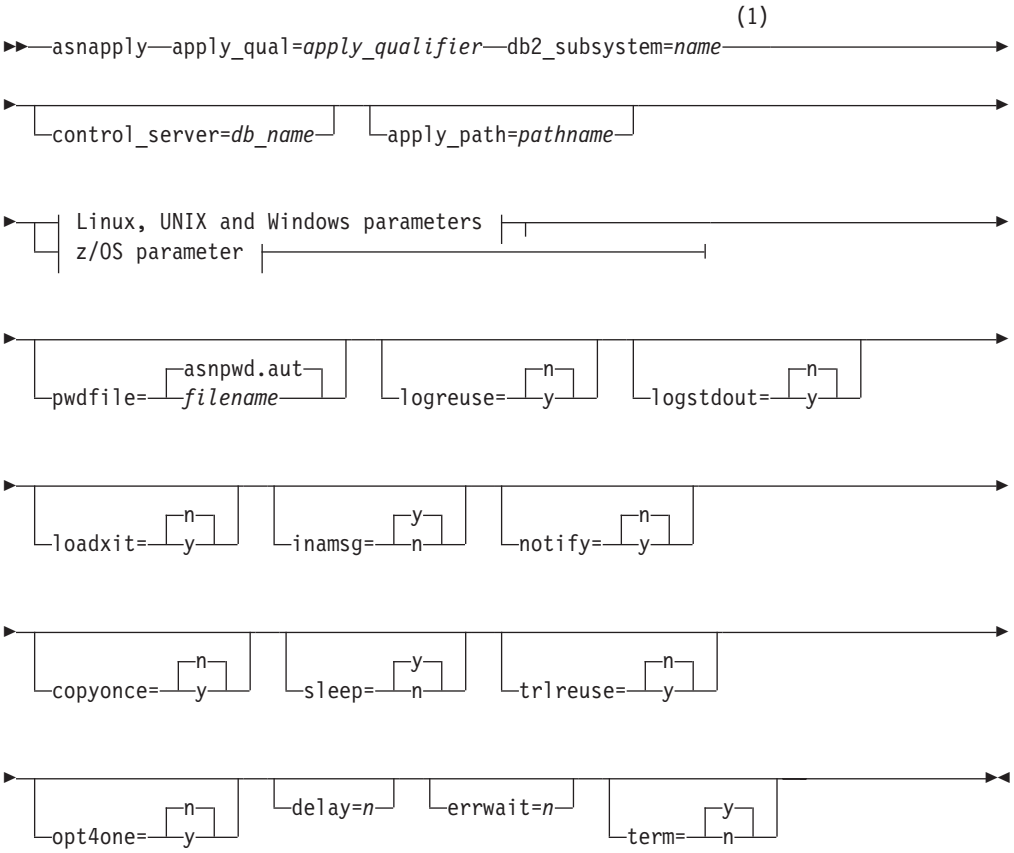
asnapply: Starting Apply

Use the **asnapply** command to start the Apply program on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. Run this command at an operating system prompt or in a shell script.

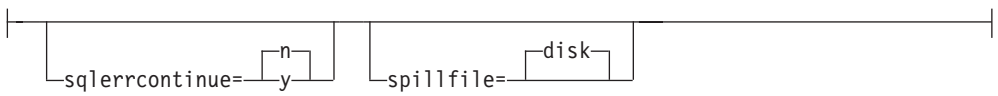
After you start the Apply program, it runs continuously until:

- You stop it in an orderly way.
- You cancel it.
- An unexpected error or failure occurs.

Syntax



UNIX, Windows and z/OS



z/OS parameter:

Notes:

1 Use the db2_subsystem parameter only on z/OS operating systems.

Parameters

Table 28 defines the invocation parameters.

Table 28. asnapply invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
apply_qual=*apply_qualifier*	Specifies the Apply qualifier that the Apply program uses to identify the subscription sets to be served. This parameter is required. The value that you enter must match the value of the APPLY_QUAL column in the subscription sets (IBMSNAP_SUBS_SET) table. The Apply qualifier name is case sensitive and can be a maximum of 18 characters.
db2_subsystem=*name*	**For z/OS only:** Specifies the name of the DB2 subsystem where the Apply program will run. The DB2 subsystem name that you enter can be a maximum of four characters. There is no default for this parameter. This parameter is required.
control_server=*db_name*	Specifies the name of the Apply control server on which the subscription definitions and Apply program control tables reside. **For Linux, UNIX and Windows:** If you do not specify an Apply control server, this parameter defaults to the value from the DB2DBDFT environment variable. **For z/OS:** Specifies the location name of the Apply control server.
apply_path=*pathname*	Specifies the location of the work files used by the Apply program. The default is the directory where the **asnapply** command was invoked.
pwdfile=*filename*	Specifies the name of the password file. If you do not specify a password file, the default is asnpwd.aut. This command searches for the password file in the directory specified by the **apply_path** parameter. If no **apply_path** parameter is specified, this command searches for the password file in the directory where the command was invoked.

Table 28. asnapply invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
logreuse=*y/n*	Specifies whether the Apply program reuses or appends messages to the log file (*db2instance.control_server.apply_qualifier.*APP.log). **n** (default) The Apply program appends messages to the log file, even after the Apply program is restarted. **y** The Apply program reuses the log file by deleting it and then re-creating it when the Apply program is restarted. **For z/OS:** The log file name does not contain the DB2 instance name (*control_server.apply_qualifier.*APP.log).
logstdout=*y/n*	Specifies where the Apply program directs the log file messages: **n** (default) The Apply program directs most log file messages to the log file only. Initialization messages go to both the log file and the standard output (STDOUT). **y** The Apply program directs log file messages to both the log file and the standard output (STDOUT).
loadxit=*y/n*	Specifies whether the Apply program invokes ASNLOAD. ASNLOAD is an IBM-supplied exit routine that uses the export and load utilities to refresh target tables. **n** (default) The Apply program does not invoke ASNLOAD. **y** The Apply program invokes ASNLOAD.
inamsg=*y/n*	Specifies whether the Apply program issues a message when the Apply program is inactive. **y** (default) The Apply program issues a message when inactive. **n** The Apply program does not issue a message when inactive.
notify=*y/n*	Specifies whether the Apply program should invoke ASNDONE. ASNDONE is an exit routine that returns control to you when the Apply program finishes copying a subscription set. **n** (default) The Apply program does not invoke ASNDONE. **y** The Apply program invokes ASNDONE.

Table 28. asnapply invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
copyonce=*y/n*	Specifies whether the Apply program executes one copy cycle for each subscription set that is eligible at the time the Apply program is invoked. Then the Apply program terminates. An eligible subscription set meets the following criteria: • (ACTIVATE > 0) in the subscription sets (IBMSNAP_SUBS_SET) table. When the ACTIVATE column value is greater than zero, the subscription set is active indefinitely or is used for a one-time-only subscription processing. • (REFRESH_TYPE = R or B) or (REFRESH_TYPE = E and the specified event occurred). The REFRESH_TYPE column value is stored in the IBMSNAP_SUBS_SET table. The MAX_SYNCH_MINUTES limit from the subscription sets table and the END_OF_PERIOD timestamp from the subscription events (IBMSNAP_SUBS_EVENT) table are honored if specified. **n** (default) The Apply program does not execute one copy cycle for each eligible subscription set. **y** The Apply program executes one copy cycle for each eligible subscription set.
sleep=*y/n*	Specifies how the Apply program is to proceed if no new subscription sets are eligible for processing. **y** (default) The Apply program goes to sleep. **n** The Apply program stops.
trlreuse=*y/n*	Specifies whether the Apply program empties the Apply trail (IBMSNAP_APPLYTRAIL) table when the Apply program starts. **n** (default) The Apply program appends entries to the IBMSNAP_APPLYTRAIL table. The Apply program does not empty the table. **y** The Apply program empties the IBMSNAP_APPLYTRAIL table during program startup.

Table 28. asnapply invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
opt4one=*y/n*	Specifies whether the performance of the Apply program is optimized if only one subscription set is defined for the Apply program. **n** (default) The performance of the Apply program is not optimized for one subscription set. **y** The performance of the Apply program is optimized for one subscription set. If you set optimization to y, the Apply program caches and reuses the information about the subscription-set members. This reuse of subscription-set member information reduces CPU usage and improves throughput rates.
delay=*n*	Specifies the delay time (in seconds) at the end of each Apply cycle when continuous replication is used, where *n*=0, 1, 2, 3, 4, 5, or 6. The default is 6, and is used during continuous replication (that is, when the subscription set uses sleep=0 minutes). This parameter is ignored if **copyonce** is specified.
errwait=*n*	Specifies the number of seconds (1 to 65535) that the Apply program waits before retrying after the program encounters an error condition. The default value is 300 seconds (five minutes). **Important:** Do not specify too small a number, because the Apply program runs almost continuously and generates many rows in the Apply trail (IBMSNAP_APPLYTRAIL) table.
term=*y/n*	Specifies how the status of DB2 affects the operation of the Apply program. **y** (default) The Apply program terminates if DB2 terminates. **n** The Apply program waits for DB2 to start, if DB2 is not active. **For Linux, UNIX and Windows:** If DB2 quiesces and the Apply program is active, the Apply program stays active and does not reconnect until DB2 is out of quiesce mode. **For z/OS:** If DB2 quiesces and the Apply program is active, the Apply program stays active and does not reconnect until DB2 is started again.

Table 28. asnapply invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
sqlerrcontinue=*y/n*	Specifies whether the Apply program continues processing when it encounters certain SQL errors. The Apply program checks the failing SQLSTATE against the values specified in the SQLSTATE file, which you create before running the Apply program. If a match is found, the Apply program writes the information about the failing row to an error file (*apply_qualifier*.ERR) and continues processing. The SQLSTATE file can contain up to 20 five-byte values. **n** (default) The Apply program does not check the SQLSTATE file. **y** The Apply program checks the SQLSTATE file during processing.
spillfile=*filetype*	Specifies where the fetched answer set is stored. **For Linux, UNIX and Windows,** valid values are: **disk** (default) A disk file. **For z/OS,** valid values are: **mem** (default) A memory file. The Apply program fails if there is insufficient memory for the answer set. **disk** A disk file.

Return codes

The **asnapply** command returns a zero return code upon successful completion. A nonzero return code is returned if the command is unsuccessful.

Examples for asnapply

The following examples illustrate how to use the **asnapply** command.

Example 1

To start an Apply program using an Apply qualifier named AQ1, a control server named dbx with work files located in the /home/files/apply/ directory:

```
asnapply apply_qual=AQ1 control_server=dbx apply_path=/home/files/apply/
  pwdfile=pass1.txt
```

The Apply program searches the /home/files/apply/ directory for the password file named pass1.txt.

Example 2

To start an Apply program that invokes the ASNLOAD exit routine:

```
asnapply apply_qual=AQ1 control_server=dbx pwdfile=pass1.txt loadxit=y
```

In this example, the Apply program searches the current directory for the password file named pass1.txt.

Example 3

To start an Apply program that executes one copy cycle for each eligible subscription set:

```
asnapply apply_qual=AQ1 control_server=dbx apply_path=/home/files/apply/
  copyonce=y
```

In this example, the Apply program searches the /home/files/apply/ directory for the default password file named asnpwd.aut.

Related tasks:

- Chapter 20, "Operating the SQL replication programs (z/OS)," on page 405

Related reference:

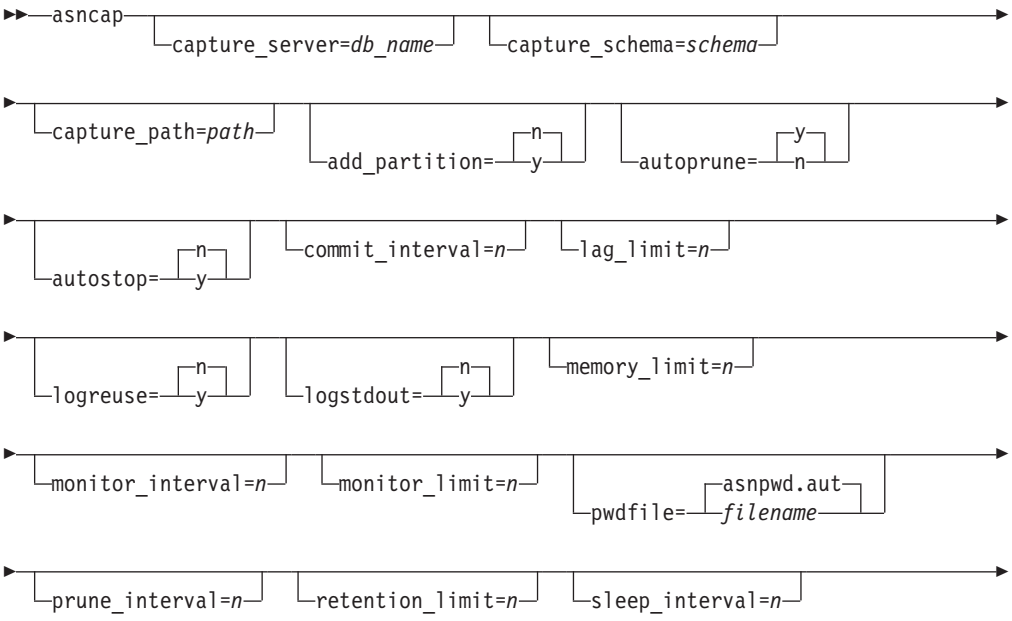
- "STRDPRAPY: Starting Apply (OS/400)" on page 386

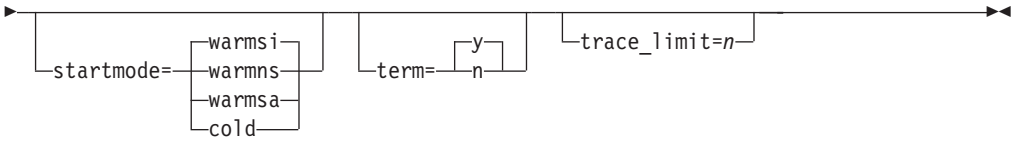
asncap: Starting Capture

Use the **asncap** command to start the Capture program on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. Run this command at an operating system prompt or in a shell script rather than through the Replication Center.

After you start the Capture program, it runs continuously until you stop it or it detects an unrecoverable error.

Syntax





Parameters

Table 29 defines the invocation parameters.

Table 29. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
capture_server=*db_name*	Specifies the name of the Capture control server. **For Linux, UNIX and Windows:** If you do not specify a Capture control server, this parameter defaults to the value from the DB2DBDFT environment variable. **For z/OS:** Specifies the name of the DB2 subsystem where the Capture program will run. For data sharing, do not use the group attach name. Instead, specify a member subsystem name.
add_partition=*y/n*	**For Linux, UNIX and Windows only:** Specifies whether the Capture program starts reading the log file for the newly added partitions since the last time the Capture program was restarted. **n** (default) No new partitions have been added since the last time the Capture program was restarted. **y** The Capture program starts reading the log file on one or more of the new partitions. On each partition, the Capture program starts reading the log from the log sequence number (LSN) that was initially used the last time the database was started.
capture_schema=*schema*	Specifies the name of the Capture schema that is used to identify a particular Capture program. The schema name that you enter must be 1 to 30 characters in length. The default is ASN.
capture_path=*path*	Specifies the location of the work files used by the Capture program. The default is the directory where the **asncap** command was invoked.

Table 29. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
autoprune=*y/n*	Specifies whether automatic pruning of the rows in the change-data (CD), unit-of-work (UOW), Capture monitor (IBMSNAP_CAPMON), Capture trace (IBMSNAP_CAPTRACE), and signal (IBMSNAP_SIGNAL) tables is enabled. **y** (default) The Capture program automatically prunes the eligible rows at the interval specified in the Capture parameters (IBMSNAP_CAPPARMS) table. The Capture program prunes the CD, UOW, and IBMSNAP_SIGNAL rows that are older than the retention limit, regardless of whether the rows have been replicated. **n** Automatic pruning is disabled.
autostop=*y/n*	Specifies whether the Capture program terminates after retrieving all the transactions that were logged before the Capture program started. **n** (default) The Capture program does not terminate after retrieving the transactions. **y** The Capture program terminates after retrieving the transactions.
commit_interval=*n*	Specifies the number of seconds that the Capture program waits before committing rows to the unit-of-work (UOW) and change-data (CD) tables. The default is 30 seconds.
lag_limit=*n*	Specifies the number of minutes that the Capture program is allowed to lag in processing log records. The default is 10080 minutes (seven days). The Capture program checks the value of this parameter only during a warm start. If this limit is exceeded, the Capture program will not start.
logreuse=*y/n*	Specifies whether the Capture program reuses or appends messages to the log file (*db2instance.capture_server.capture_schema.*CAP.log). **n** (default) The Capture program appends messages to the log file, even after the Capture program is restarted. **y** The Capture program reuses the log file by first truncating the current log file and then starting a new log when the Capture program is restarted. **For z/OS:** The log file name does not contain the DB2 instance name (*capture_server.capture_schema.*CAP.log).

Table 29. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
logstdout=*y/n*	Specifies where the Capture program directs the log file messages: **n** (default) The Capture program directs most log file messages to the log file only. Initialization messages go to both the log file and the standard output (STDOUT). **y** The Capture program directs log file messages to both the log file and the standard output (STDOUT).
memory_limit=*n*	Specifies the maximum size (in megabytes) of memory that the Capture program can use to build transactions. After reaching this memory limit, the Capture program spills transactions to a file. The default is 32 megabytes.
monitor_interval=*n*	Specifies how frequently (in seconds) the Capture program inserts rows into the Capture monitor (IBMSNAP_CAPMON) table. The default is 300 seconds (five minutes).
monitor_limit=*n*	Specifies how long (in minutes) a row can remain in the Capture monitor (IBMSNAP_CAPMON) table before it becomes eligible for pruning. All IBMSNAP_CAPMON rows that are older than the value of the **monitor_limit** parameter are pruned at the next pruning cycle. The default is 10 080 minutes (seven days).
pwdfile=*filename*	Specifies the name of the password file. If you do not specify a password file, the default is asnpwd.aut. This command searches for the password file in the directory specified by the **capture_path** parameter. If no **capture_path** parameter is specified, this command searches for the password file in the directory where the command was invoked.
prune_interval=*n*	Specifies how frequently (in seconds) the change-data (CD), unit-of-work (UOW), Capture monitor (IBMSNAP_CAPMON), Capture trace (IBMSNAP_CAPTRACE), and signal (IBMSNAP_SIGNAL) tables are pruned. This parameter is ignored if you set the autoprune parameter to `n`. The default is 300 seconds (five minutes).
retention_limit=*n*	Specifies how long (in minutes) a row can remain in the change-data (CD), unit-of-work (UOW), or signal (IBMSNAP_SIGNAL) table before it becomes eligible for pruning. Each row that is older than the value of the **retention_limit** parameter is pruned at the next pruning cycle. The default is 10 080 minutes (seven days).
sleep_interval=*n*	Specifies the number of seconds that the Capture program sleeps when it finishes processing the active log and determines that the buffer is empty. The default is five seconds. **For z/OS:** Specifies the number of seconds that the Capture program sleeps after the buffer returns less than half full.

Table 29. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
startmode=*mode*	Specifies the processing procedure used by the Capture program during a Capture startup. **warmsi** (default) The Capture program resumes processing where it ended in its previous run if warm start information is available. If this is the first time that you are starting the Capture program, it automatically switches to a cold start. During a warm start, the Capture program leaves the Capture trace (IBMSNAP_CAPTRACE), change-data (CD), unit-of-work (UOW), and restart (IBMSNAP_RESTART) tables intact. If errors occur after the Capture program started, the Capture program terminates. **warmns** The Capture program resumes processing where it ended in its previous run if warm start information is available. If errors occur after the Capture program started, the Capture program terminates. If the Capture program cannot warm start, it *does not* switch to a cold start. **warmsa** The Capture program resumes processing where it ended in its previous run if warm start information is available. If the Capture program cannot warm start, it switches to a cold start. **cold** The Capture program starts by deleting all rows in its CD and UOW tables. Most registrations are reset so that all subscriptions to those sources are fully refreshed during the next Apply processing cycle. Registrations for external CCDs and those subscriptions whose targets are noncomplete CCDs are not fully refreshed.
term=*y/n*	Specifies whether the Capture program terminates if DB2 terminates. **y** (default) The Capture program terminates if DB2 terminates. **n** The Capture program continues running if DB2 terminates with MODE(QUIESCE). When DB2 initializes, the Capture program starts in warm mode and begins capturing at the point it left off when DB2 terminated. If DB2 terminates via FORCE or due to abnormal termination, the Capture program terminates even if you set this parameter to `n`. If you set this parameter to `n` and start DB2 with restricted access (ACCESS MAINT), the Capture program cannot connect and subsequently terminates.

Table 29. asncap invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
trace_limit=*n*	Specifies how long (in minutes) a row can remain in the Capture trace (IBMSNAP_CAPTRACE) table before it becomes eligible for pruning. All IBMSNAP_CAPTRACE rows that are older than the value of the **trace_limit** parameter are pruned at the next pruning cycle. The default is 10 080 minutes (seven days).

Return codes

The **asncap** command returns a zero return code upon successful completion. A nonzero return code is returned if the command is unsuccessful.

Examples for asncap

The following examples illustrate how to use the **asncap** command.

Example 1

To start a Capture program for the first time using a Capture control server named db and a Capture schema of ASN with work files located in the /home/files/capture/logs/ directory:

```
asncap capture_server=db capture_schema=ASN
  capture_path=/home/files/capture/logs/ startmode=cold
```

Example 2

To restart a Capture program without pruning after the Capture program was stopped:

```
asncap capture_server=db autoprune=n sleep_interval=10 startmode=warmsa
```

The Capture program in this example retains all rows in the corresponding control tables and sleeps for ten seconds after it finishes processing the active log and determines that the buffer is empty. The Capture program resumes processing where it ended in its previous run and switches to a cold start if warm start information is unavailable.

Example 3

To restart a Capture program with the warmns startmode and changed parameter settings:

```
asncap capture_server=db autoprune=y prune_interval=60 retention_limit=1440
  startmode=warmns
```

This command restarts the Capture program and uses new parameter settings to decrease the amount of time before the CD, UOW, and IBMSNAP_SIGNAL tables become eligible for pruning and to increase the frequency of pruning from the default parameter settings. The Capture program resumes processing where it ended in its previous run but does not automatically switch to a cold start if warm start information is unavailable.

Example 4

To start a Capture program that sends all of its work files to a new subdirectory called capture_files:

1. Go to the appropriate directory, and then create a new subdirectory called capture_files:

```
cd /home/db2inst
mkdir capture_files
```

2. Start the Capture program, and specify a Capture path that is located in the new subdirectory that you just created:

```
asncap capture_server=db capture_schema=ASN
  capture_path=/home/db2inst/capture_files startmode=warmsi
```

Related tasks:

- Chapter 20, "Operating the SQL replication programs (z/OS)," on page 405

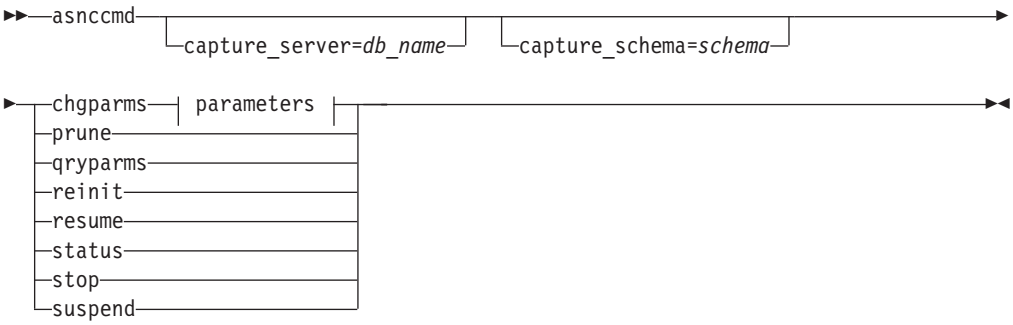
Related reference:

- "STRDPRCAP: Starting Capture (OS/400)" on page 393

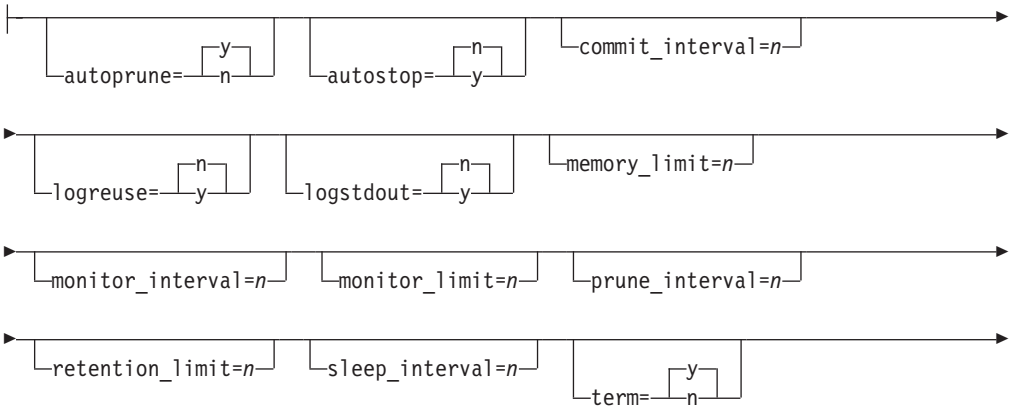
asnccmd: Operating Capture

Use the **asnccmd** command to operate the Capture program on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. Run this command at an operating system prompt or in a shell script.

Syntax



Parameters:



Parameters

The invocation parameters for this command are identical to those in "asncap: Starting Capture" on page 282. See Table 29 on page 283 for definition of those parameters.

Table 30 defines the chgparms invocation parameters.

Table 30. asnccmd chgparms parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
autoprune=*y/n*	Specifies whether automatic pruning of the rows in the change-data (CD), unit-of-work (UOW), Capture monitor (IBMSNAP_CAPMON), Capture trace (IBMSNAP_CAPTRACE), and signal (IBMSNAP_SIGNAL) tables is enabled. **y** (default) The Capture program automatically prunes the eligible rows at the interval specified in the Capture parameters (IBMSNAP_CAPPARMS) table. The Capture program prunes the CD, UOW, and IBMSNAP_SIGNAL rows that are older than the retention limit, regardless of whether the rows have been replicated. **n** Automatic pruning is disabled.
autostop=*y/n*	Specifies whether the Capture program terminates after retrieving all the transactions logged before the Capture program started. **n** (default) The Capture program does not terminate after retrieving the transactions. **y** The Capture program terminates after retrieving the transactions.
commit_interval=*n*	Specifies the number of seconds that the Capture program waits before committing rows to the unit-of-work (UOW) and change-data (CD) tables. The default is 30 seconds.

Table 30. asnccmd chgparms parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
logreuse=*y/n*	Specifies whether the Capture program reuses or appends messages to the log file (*db2instance.capture_server.capture_schema.*CAP.log). **n** (default) The Capture program appends messages to the log file, even after the Capture program is restarted. **y** The Capture program reuses the log file by first truncating the current log file and then starting a new log when the Capture program is restarted. If you change this parameter to *y* through the use of the **chgparms** parameter, the log is immediately truncated and reused. This change to the **logreuse** parameter does not affect the next start of the Capture program. **For z/OS:** The log file name does not contain the DB2 instance name (*capture_server.capture_schema.*CAP.log).
logstdout=*y/n*	Specifies where messages are sent by the Capture program. **n** (default) The Capture program sends messages to the log file only. **y** The Capture program sends messages to both the log file and the standard output (stdout).
memory_limit=*n*	Specifies the maximum size (in megabytes) of memory that the Capture program can use to build transactions. After reaching this memory limit, the Capture program spills transactions to a file. The default is 32 megabytes.
monitor_interval=*n*	Specifies how frequently (in seconds) the Capture program inserts rows into the Capture monitor (IBMSNAP_CAPMON) table. The default is 300 seconds (five minutes).
monitor_limit=*n*	Specifies how long (in minutes) a row can remain in the Capture monitor (IBMSNAP_CAPMON) table before it becomes eligible for pruning. All IBMSNAP_CAPMON rows that are older than the value of the **monitor_limit** parameter are pruned at the next pruning cycle. The default is 10 080 minutes (seven days).
prune_interval=*n*	Specifies how frequently (in seconds) the change-data (CD), unit-of-work (UOW), Capture monitor (IBMSNAP_CAPMON), Capture trace (IBMSNAP_CAPTRACE), and signal (IBMSNAP_SIGNAL) tables are pruned. This parameter is ignored if you set the autoprune parameter to n. The default is 300 seconds (five minutes).
retention_limit=*n*	Specifies how long (in minutes) a row can remain in the change-data (CD), unit-of-work (UOW), or signal (IBMSNAP_SIGNAL) table before it becomes eligible for pruning. Each row that is older than the value of the **retention_limit** parameter is pruned at the next pruning cycle. The default is 10 080 minutes (seven days).

Table 30. asnccmd chgparms parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
sleep_interval=*n*	Specifies the number of seconds that the Capture program sleeps when it finishes processing the active log and determines that the buffer is empty. The default is five seconds. **For z/OS:** Specifies the number of seconds that the Capture program sleeps after the buffer returns less than half full.
term=*y*/*n*	Specifies whether the Capture program terminates if DB2 terminates. **y** (default) The Capture program terminates if DB2 terminates. **n** The Capture program continues running if DB2 terminates with MODE(QUIESCE). When DB2 initializes, the Capture program starts in warm mode and begins capturing at the point it left off when DB2 terminated. If DB2 terminates via FORCE or due to an abnormal termination, the Capture program terminates even if you set this parameter to `n`. If you set this parameter to `n` and start DB2 with restricted access (ACCESS MAINT), the Capture program cannot connect and subsequently terminates.
trace_limit=*n*	Specifies how long (in minutes) a row can remain in the Capture trace (IBMSNAP_CAPTRACE) table before it becomes eligible for pruning. All IBMSNAP_CAPTRACE rows that are older than the value of the **trace_limit** parameter are pruned at the next pruning cycle. The default is 10 080 minutes (seven days).

Examples for asnccmd

The following examples illustrate how to use the **asnccmd** command.

Example 1

To enable a running Capture program to recognize newly added replication sources:

```
asnccmd capture_server=db capture_schema=ASN reinit
```

Example 2

To prune the CD, UOW, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_SIGNAL tables once:

```
asnccmd capture_server=db capture_schema=ASN prune
```

Example 3

To receive messages about the state of each Capture thread:

```
asnccmd capture_server=db capture_schema=ASN status
```

Example 4

To send the current operational values of a Capture program to the standard output:

```
asnccmd capture_server=db capture_schema=ASN qryparms
```

Example 5

To disable the automatic pruning in a running Capture program:

```
asnccmd capture_server=db capture_schema=ASN chgparms autoprune=n
```

Example 6

To stop a running Capture program:

```
asnccmd capture_server=db capture_schema=ASN stop
```

Related tasks:

- Chapter 20, "Operating the SQL replication programs (z/OS)," on page 405

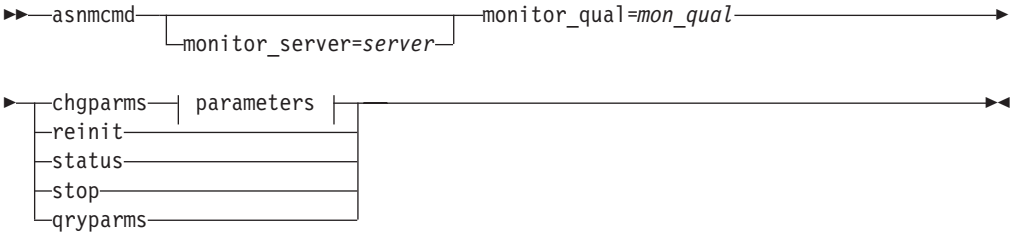
Related reference:

- "OVRDPRCAPA: Overriding DPR capture attributes (OS/400)" on page 375

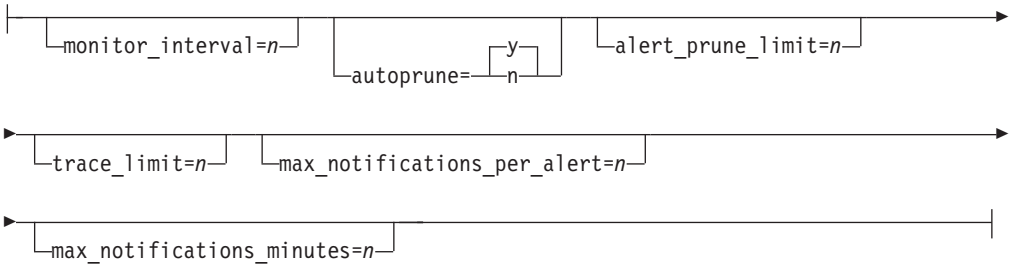
asnmcmd: Working with a running Replication Alert Monitor

Use **asnmcmd** to send commands to a running Replication Alert Monitor on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. Run this command at an operating system prompt or in a shell script.

Syntax



Parameters:



Parameters

Table 31 defines the invocation parameters for the **asnmcmd** command.

Table 31. asnmcmd invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
monitor_server=*server*	Specifies the name of the Monitor control server where the Replication Alert Monitor program runs and the monitor control tables reside. This must be the first parameter if entered. **Linux, UNIX, Windows:** If you do not specify a Monitor control server, this parameter defaults to the value from the DB2DBDFT environment variable. **z/OS:** The default is DSN.
monitor_qual=*mon_qual*	Specifies the monitor qualifier that the Replication Alert Monitor program uses. The monitor qualifier identifies the server to be monitored and the associated monitoring conditions. You must specify a monitor qualifier. The monitor qualifier name is case sensitive and can be a maximum of 18 characters.
chgparms	Specify to change one or more of the following operational parameters of the Replication Alert Monitor while it is running: • **monitor_interval** • **autoprune** • **alert_prune_limit** • **trace_limit** • **max_notifications_per_alert** • **max_notifications_minutes** You can specify multiple parameters in one **chgparms** subcommand, and you can change these parameter values as often as you want. The changes temporarily override the values in the IBMSNAP_MONPARMS table, but they are not saved in the table. When you stop and restart the Replication Alert Monitor, it uses the values in IBMSNAP_MONPARMS. asnmon: Starting a Replication Alert Monitor includes descriptions of the parameters that you can override with this subcommand. **Important:** The parameter that you are changing must immediately follow the **chgparms** subcommand.
reinit	Specify to have the Replication Alert Monitor program read its control tables to refresh the data that it has for contacts, alert conditions, and parameters in its memory. When all values are read, the Monitor program begins its cycle of checking conditions on the servers. After this cycle is complete, the next monitor cycle begins after the time specified in **monitor_interval** has elapsed.
status	Specify to receive messages that indicate the state of each thread (administration, serialization, and worker) in the Replication Alert Monitor.

Table 31. asnmcmd invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
qryparms	Specify if you want the current operational parameter values for the Replication Alert Monitor written to the standard output (stdout).
stop	Specify to stop the Replication Alert Monitor in an orderly way.

Examples for asnmcmd

The following examples illustrate how to use the **asnmcmd** command.

Example 1

To stop the Replication Alert Monitor for the specified monitor qualifier:

```
asnmcmd monitor_server=wsdb monitor_qual=monqual stop
```

Example 2

To receive messages that indicate the status of the Replication Alert Monitor threads:

```
asnmcmd monitor_server=wsdb monitor_qual=monqual status
```

Example 3

To refresh the Replication Alert Monitor with current values from the monitor control tables:

```
asnmcmd monitor_server=wsdb monitor_qual=monqual reinit
```

Example 4

To reduce the maximum number of notifications that the Replication Alert Monitor sends during a specified time period from the default of 3:

```
asnmcmd monitor_server=wsdb monitor_qual=monqual
chgparms max_notifications_per_alert=2
```

Example 5

To send the current operational parameters of the Replication Alert Monitor to the standard output:

```
asnmcmd monitor_server=wsdb monitor_qual=monqual qryparms
```

Related reference:

- "asnmon: Starting a Replication Alert Monitor" on page 295

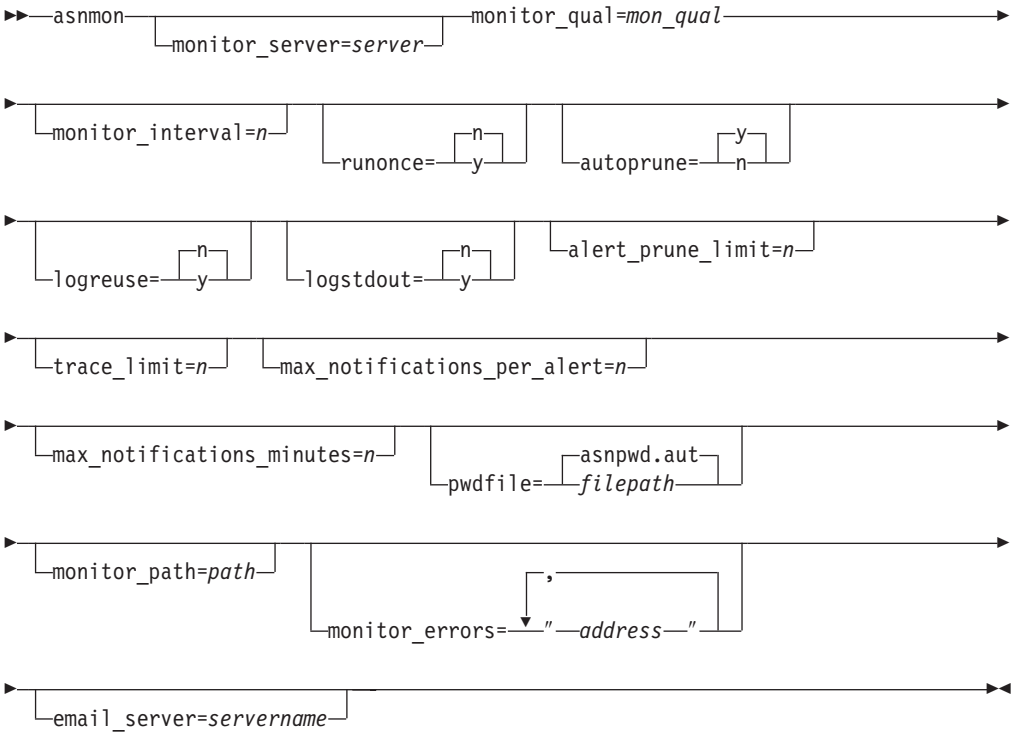
asnmon: Starting a Replication Alert Monitor

Use the **asnmon** command to start a Replication Alert Monitor on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. Run this command at an operating system prompt or in a shell script.

The Replication Alert Monitor records the following information:

- The status of Q capture and Q Apply programs, and Capture and Apply programs
- Error messages written to the control tables
- Threshold values

Syntax



Parameters

Table 32 defines the invocation parameters for the **asnmon** command.

Table 32. asnmon invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
monitor_server=*server*	Specifies the name of the Monitor control server where the Replication Alert Monitor program runs and the monitor control tables reside. This must be the first parameter if entered. **Linux, UNIX, Windows:** If you do not specify a Monitor control server, this parameter defaults to the value from the DB2DBDFT environment variable. **z/OS:** The default is DSN.

Table 32. asnmon invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
monitor_qual=*mon_qual*	Specifies the monitor qualifier that the Replication Alert Monitor program uses. The monitor qualifier identifies the server to be monitored and the associated monitoring conditions. You must specify a monitor qualifier. The monitor qualifier name is case sensitive and can be a maximum of 18 characters.
monitor_interval=*n*	Specifies how frequently (in seconds) the Replication Alert Monitor program runs for this monitor qualifier. The default is 300 seconds (five minutes). This parameter is ignored by the Replication Alert Monitor if you set the **runonce** parameter to y. **Important:** This **monitor_interval** parameter affects the Replication Alert Monitor program only. This parameter does not affect Q Capture, Q Apply, Capture, and Apply programs.
runonce=**y/n**	Specifies whether the Replication Alert Monitor program runs only one time for this monitor qualifier. **n** (default) The Replication Alert Monitor program runs at the frequency indicated by the **monitor_interval** parameter. **y** The Replication Alert Monitor program runs only one monitor cycle. If you set the runonce parameter to y, the **monitor_interval** parameter is ignored by the Replication Alert Monitor.
autoprune=**y/n**	Specifies whether automatic pruning of the rows in the Replication Alert Monitor alerts (IBMSNAP_ALERTS) table is enabled. **y** (default) The Replication Alert Monitor program automatically prunes the rows in the IBMSNAP_ALERTS table that are older than the value of the **alert_prune_limit** parameter. **n** Automatic pruning is disabled.
logreuse=**y/n**	Specifies whether the Replication Alert Monitor program reuses or appends messages to its diagnostic log file (*db2instance.monitor_server.mon_qual*.MON.log). **n** (default) The Replication Alert Monitor program appends messages to the log file. **y** The Replication Alert Monitor program reuses the log file by deleting it and then re-creating it when the Replication Alert Monitor program is restarted.

Table 32. asnmon invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
logstdout=**y/n**	Specifies where messages are sent by the Replication Alert Monitor program. **n** (default) The Replication Alert Monitor program sends messages to the log file only. **y** The Replication Alert Monitor program sends messages to both the log file and the standard output (stdout).
alert_prune_limit=*n*	Specifies how long (in minutes) rows are kept in the Replication Alert Monitor alerts (IBMSNAP_ALERTS) table. Any rows older than this value are pruned. The default is 10 080 minutes (seven days).
trace_limit=*n*	Specifies how long (in minutes) a row can remain in the Replication Alert Monitor trace (IBMSNAP_MONTRACE) table before it becomes eligible for pruning. All IBMSNAP_MONTRACE rows that are older than the value of this **trace_limit** parameter are pruned at the next pruning cycle. The default is 10 080 minutes (seven days).
max_notifications_per_alert=*n*	Specifies the maximum number of the same alerts that are sent to a user when the alerts occurred during the time period specified by the **max_notifications_minutes** parameter value. Use this parameter to avoid re-sending the same alerts to a user. The default is 3.
max_notifications_minutes=*n*	This parameter works with the **max_notifications_per_alert** parameter to indicate the time period when alert conditions occurred. The default is 60 minutes.
pwdfile=*filepath*	Specifies the fully qualified name of the password file. You define this file using the **asnpwd** command. The default file name is asnpwd.aut.
monitor_path=*path*	Specifies the location of the log files used by the Replication Alert Monitor program. The default is the directory where the **asnmon** command was invoked.
monitor_errors=*address*	Specifies the e-mail address to which notifications are sent if a fatal error is detected before the alert monitor connects to the Monitor control server. Use this parameter to send a notification that the Monitor control server connection failed because of invalid start parameters, an incorrect monitor qualifier, a down database, or other error. Type double quotation marks around the e-mail address text. You can enter multiple e-mail addresses. Separate the e-mail addresses with commas. You can type spaces before or after the commas.
email_server=*servername*	Specifies the e-mail server address. Enter this parameter *only* if you use the ASNMAIL exit routine with SMTP (Simple Mail Transfer Protocol).

Return codes

The **asnmon** command returns a zero return code upon successful completion. A nonzero return code is returned if the command is unsuccessful.

Examples for asnmon

The following examples illustrate how to use the **asnmon** command.

Example 1

To start the Replication Alert Monitor with the default parameters:

```
asnmon monitor_server=wsdb monitor_qual=monqual
```

Example 2

To start a Replication Alert Monitor that runs every 120 seconds (two minutes) for the specified monitor qualifier:

```
asnmon monitor_server=wsdb monitor_qual=monqual monitor_interval=120
```

Example 3

To start a Replication Alert Monitor and specify that it run only once for the specified monitor qualifier:

```
asnmon monitor_server=wsdb monitor_qual=monqual runonce=y
```

Example 4

To start a Replication Alert Monitor that sends e-mail notifications if it detects monitoring errors:

```
asnmon monitor_server=wsdb monitor_qual=monqual
  monitor_errors="repladm@company.com, dbadmin@company.com"
```

Example 5

To start a Replication Alert Monitor that runs every 120 seconds (two minutes) and waits 1440 minutes (24 hours) before sending alerts:

```
asnmon monitor_server=wsdb monitor_qual=monqual monitor_interval=120
  max_notifications_per_alert=2 max_notifications_minutes=1440
```

This Replication Alert Monitor program sends a maximum of two alerts when the alerts occurred during the time period specified by the **max_notifications_minutes** parameter value (1440 minutes).

Related reference:

- "asnmcmd: Working with a running Replication Alert Monitor" on page 292

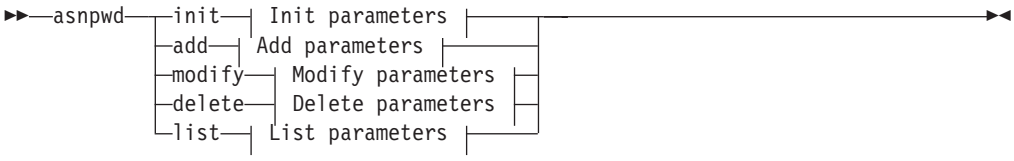
asnpwd: Creating and maintaining password files

Use the **asnpwd** command to create and change password files on Linux, UNIX, and Windows. Run this command at the command line or in a shell script.

The parameter keywords of this command are *not* case sensitive.

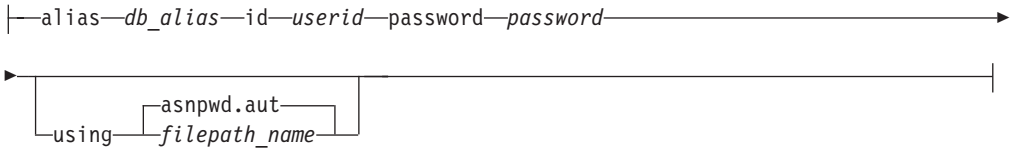
Command help appears if you enter the **asnpwd** command without any parameters, followed by a *?*, or followed by incorrect parameters.

Syntax

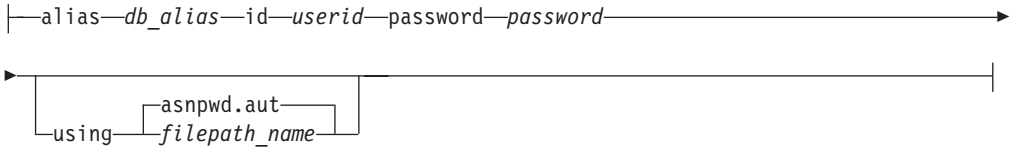


Init parameters:

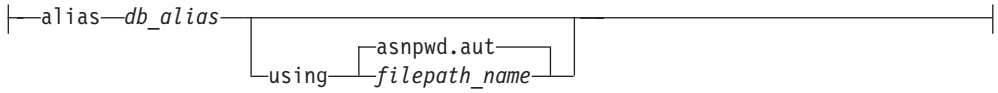
Add parameters:



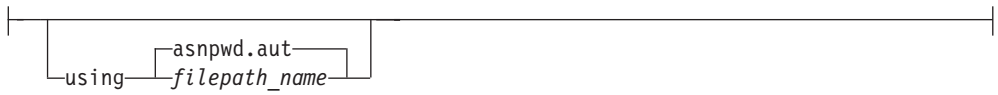
Modify parameters:



Delete parameters:



List parameters:



Parameters

Table 33 defines the invocation parameters for the **asnpwd** command.

Table 33. asnpwd invocation parameter definitions for Linux, UNIX, and Windows operating systems

Parameter	Definition
init	Specify to create an empty password file. This command will fail if you specify the **init** parameter with a password file that already exists.
add	Specify to add an entry to the password file. This command will fail if you specify the **add** parameter with an entry that already exists in the password file. Use the **modify** parameter to change an existing entry in the password file.
modify	Specify to modify the password or user ID for an entry in the password file.
delete	Specify to delete an entry from the password file.
list	Specify to list the aliases and user ID entries in a password file. This parameter can be used only if the password file was created using the **encrypt** parameter. Passwords are never displayed by the **list** command.
encrypt	Specifies which entries in a file to encrypt. **all** (default) Encrypt all entries in the specified file such that you cannot list the database aliases, user names, and passwords that are in the file. This option reduces the exposure of information in password files. **password** Encrypt the password entry in the specified file. This option allows users to list the database aliases and user names stored in their password file. Passwords can never be displayed.
using *filepath_name*	Specifies the path and name of the password file. Follow the file naming conventions of your operating system. An example of a valid password file on Windows is C:\sqllib\mypwd.aut. If you specify the path and name of the password file, the path and the password file must already exist. If you are using the **init** parameter and you specify the path and name of the password file, the path must already exist and the command will create the password file for you. If you do not specify this parameter, the default file name is asnpwd.aut and the default file path is the current directory.
alias *db_alias*	Specifies the alias of the database to which the user ID has access. The alias is always folded to uppercase, regardless of how it is entered.
id *userid*	Specifies the user ID that has access to the database.
password *password*	Specifies the password for the specified user ID. This password is case sensitive and is encrypted in the password file.

Return codes

The **asnpwd** command returns a zero return code upon successful completion. A nonzero return code is returned if the command is unsuccessful.

Examples for asnpwd

The following examples illustrate how to use the **asnpwd** command.

Example 1

To create a password file with the default name of asnpwd.aut in the current directory:

```
asnpwd INIT
```

Example 2

To create a password file named pass1.aut in the c:\myfiles directory:

```
asnpwd INIT USING c:\myfiles\pass1.aut
```

Example 3

To create a password file named mypwd.aut using the **encrypt all** parameter:

```
asnpwd INIT ENCRYPT ALL USING mypwd.aut
```

Example 4

To create a password file named mypwd.aut using the **encrypt password** parameter:

```
asnpwd INIT ENCRYPT PASSWORD USING mypwd.aut
```

Example 5

To create a default password file using the **encrypt password** parameter:

```
asnpwd INIT ENCRYPT PASSWORD
```

Example 6

To add a user ID called oneuser and its password to the password file named pass1.aut in the c:\myfiles directory and to grant this user ID access to the db1 database:

```
asnpwd ADD ALIAS db1 ID oneuser PASSWORD mypwd using c:\myfiles\pass1.aut
```

Example 7

To modify the user ID or password of an entry in the password file named pass1.aut in the c:\myfiles directory:

```
asnpwd MODIFY AliaS sample ID chglocalid PASSWORD chgmajorpwd
  USING c:\myfiles\pass1.aut
```

Example 8

To delete the database alias called sample from the password file named pass1.aut in the c:\myfiles directory:

```
asnpwd delete alias sample USING c:\myfiles\pass1.aut
```

Example 9

To see command help:

```
asnpwd
```

Example 10

To list the entries in a default password file:

```
asnpwd LIST
```

Example 11

To list the entries in a password file named pass1.aut:

```
asnpwd LIST USING pass1.aut
```

The output from this command depends on how the password file was initialized:

- If it was initialized using the **encrypt all** parameter, the following message is issued:

  ```
  ASN1986E "Asnpwd" : "". The password file "pass1.aut" contains
  encrypted information that cannot be listed.
  ```

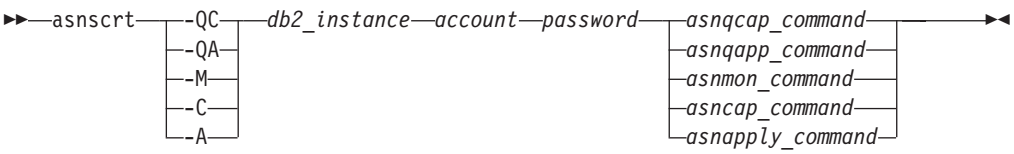
- If it was not initialized using the **encrypt all** parameter, the following details are listed:

  ```
  asnpwd LIST USING pass1.aut
  Alias: SAMPLE  ID: chglocalid
  Number of Entries: 1
  ```

asnscrt: Creating a DB2 replication service to start the replication programs

Use the **asnscrt** command to create a DB2 replication service in the Windows Service Control Manager (SCM) and invoke the **asnqcap**, **asnqapp**, **asnmon**, **asncap**, and **asnapply** commands. Run the **asnscrt** command on the Windows operating system.

Syntax



Parameters

Table 34 defines the invocation parameters for the **asnscrt** command.

Table 34. asnscrt invocation parameter definitions for Windows operating systems

Parameter	Definition
-QC	Specifies that you are starting a Q capture program.
-QA	Specifies that you are starting a Q apply program.
-M	Specifies that you are starting a Replication Alert Monitor program.
-C	Specifies that you are starting a Capture program.
-A	Specifies that you are starting an Apply program.
db2_instance	Specifies the DB2 instance used to identify a unique DB2 replication service. The DB2 instance can be a maximum of eight characters.
account	Specifies the account name that you use to log on to Windows. The account name must begin with a period and a backslash (.\).
password	Specifies the password used with the account name. If the password contains special characters, type a backslash (\) before each special character.
asnqcap_command	Specifies the complete **asnqcap** command to start a Q capture program. Use the documented **asnqcap** command syntax with the appropriate **asnqcap** parameters. **Important:** If the DB2PATH environment variable is not defined, you must specify a location for the work files by including the **capture_path** parameter with the **asnqcap** command. If the DB2PATH variable is defined and you specify a **capture_path**, the **capture_path** parameter overrides the DB2PATH variable. The **asnscrt** command does not validate the syntax of the **asnqcap** parameters that you enter.
asnqapp_command	Specifies the complete **asnqapp** command to start a Q apply program. Use the documented **asnqapp** command syntax with the appropriate **asnqapp** parameters. **Important:** If the DB2PATH environment variable is not defined, you must specify the location for the work files by including the **apply_path** parameter with the **asnqapp** command. If the DB2PATH variable is defined and you specify an **apply_path**, the **apply_path** parameter overrides the DB2PATH variable. The **asnscrt** command does not validate the syntax of the **asnqapp** parameters that you enter.

Table 34. asnscrt invocation parameter definitions for Windows operating systems (continued)

Parameter	Definition
asnmon_command	Specifies the complete **asnmon** command to start a Replication Alert Monitor program. Use the documented **asnmon** command syntax with the appropriate **asnmon** parameters. **Important:** If the DB2PATH environment variable is not defined, you must specify a location for the log files by including the **monitor_path** parameter with the **asnmon** command. If the DB2PATH variable is defined and you specify a **monitor_path**, the **monitor_path** parameter overrides the DB2PATH variable. The **asnscrt** command does not validate the syntax of the **asnmon** parameters that you enter.
asncap_command	Specifies the complete **asncap** command to start a Capture program. Use the documented **asncap** command syntax with the appropriate **asncap** parameters. **Important:** If the DB2PATH environment variable is not defined, you must specify a location for the work files by including the **capture_path** parameter with the **asncap** command. If the DB2PATH variable is defined and you specify a **capture_path**, the **capture_path** parameter overrides the DB2PATH variable. The **asnscrt** command does not validate the syntax of the **asncap** parameters that you enter.
asnapply_command	Specifies the complete **asnapply** command to start an Apply program. Use the documented **asnapply** command syntax with the appropriate **asnapply** parameters. **Important:** If the DB2PATH environment variable is not defined, you must specify the location for the work files by including the **apply_path** parameter with the **asnapply** command. If the DB2PATH variable is defined and you specify an **apply_path**, the **apply_path** parameter overrides the DB2PATH variable. The **asnscrt** command does not validate the syntax of the **asnapply** parameters that you enter.

Examples for asnscrt

The following examples illustrate how to use the **asnscrt** command.

Example 1

To create a DB2 replication service that invokes a Q capture program under a DB2 instance called inst1:

```
asnscrt -QC inst1 .\joesmith password asnqcap capture_server=mydb1
  capture_schema=QC1 capture_path=X:\logfiles
```

Example 2

To create a DB2 replication service that invokes a Q apply program under a DB2 instance called inst2 using a logon account of .\joesmith and a password of my$pwd:

```
asnscrt -QA inst2 .\joesmith my\$pwd asnqapp apply_server=mydb2 apply_schema =as2
  apply_path=X:\sqllib
```

Example 3

To create a DB2 replication service that invokes a Capture program under a DB2 instance called inst1:

```
asnscrt -C inst1 .\joesmith password asncap capture_server=sampledb
  capture_schema=ASN capture_path=X:\logfiles
```

Example 4

To create a DB2 replication service that invokes an Apply program under a DB2 instance called inst2 using a logon account of .\joesmith and a password of my$pwd:

```
asnscrt -A inst2 .\joesmith my\$pwd asnapply control_server=db2 apply_qual=aq2
  apply_path=X:\sqllib
```

Example 5

To create a DB2 replication service that invokes a Replication Alert Monitor program under a DB2 instance called inst3:

```
asnscrt -M inst3 .\joesmith password asnmon monitor_server=db3 monitor_qual=mq3
  monitor_path=X:\logfiles
```

Example 6

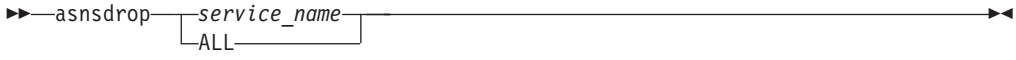
To create a DB2 replication service that invokes a Capture program under a DB2 instance called inst4 and overrides the default work file directory with a fully qualified **capture_path**:

```
asnscrt -C inst4 .\joesmith password X:\sqllib\bin\asncap capture_server=scdb
  capture_schema=ASN capture_path=X:\logfiles
```

asnsdrop: Dropping DB2 replication services

Use the **asnsdrop** command to drop DB2 replication services from the Windows Service Control Manager (SCM) on the Windows operating system. (You create a DB2 replication service using the **asnscrt** command.)

Syntax



Parameters

Table 35 defines the invocation parameters for the **asnsdrop** command.

Table 35. asnsdrop invocation parameter definitions for Windows operating systems

Parameter	Definition
service_name	Specifies the fully qualified name of the DB2 replication service. Enter the Windows SCM to obtain the DB2 replication service name. On Windows operating systems, you can obtain the service name by opening the Properties window of the DB2 replication service. If the DB2 replication service name contains spaces, enclose the entire service name in double quotation marks.
ALL	Specifies that you want to drop all DB2 replication services.

Examples for asnsdrop

The following examples illustrate how to use the **asnsdrop** command.

Example 1

To drop a DB2 replication service:

```
asnsdrop DB2.SAMPLEDB.SAMPLEDB.CAP.ASN
```

Example 2

To drop a DB2 replication service with a schema named A S N:

```
asnsdrop "DB2.SAMPLEDB.SAMPLEDB.CAP.A S N"
```

Example 3

To drop all DB2 replication services:

```
asnsdrop ALL
```

asnslist: Listing DB2 replication services

Use the **asnslist** command to list the DB2 replication services in the Windows Service Control Manager (SCM). You can optionally use the command to list details about each service. Run the **asnslist** command on the Windows operating system.

Syntax

Parameters

Table 36 defines the invocation parameter for the **asnslist** command.

Table 36. asnslist invocation parameter definition for Windows operating systems

Parameter	Definition
details	Specifies that you want to list detailed data about all DB2 replication services on a system.

Examples for asnlist

The following examples illustrate how to use the **asnslist** command.

Example 1

To list the names of DB2 replication services on a system:

```
asnslist
```

Here is an example of the command output:

```
DB2.DB2.SAMPLE.QAPP.ASN
DB2.DB4.SAMPLE.QCAP.ASN
```

Example 2

To list details about all services on a system:

```
asnslist details
```

Here is an example of the command output:

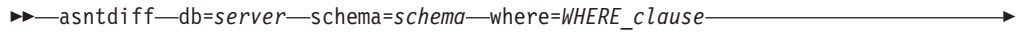
```
DB2.DB2.SAMPLE.QAPP.ASN
Display Name: DB2 DB2 SAMPLE QAPPLY ASN
Image Path:   ASNSERV DB2.DB2.SAMPLE.APP.AQ1 -ASNQAPPLY QAPPLY_SERVER=SAMPLE AP
              PLY_SCHEMA=ASN QAPPLY_PATH=C:\PROGRA~1\SQLLIB
Dependency:   DB2-0

DB2.DB4.SAMPLE.QCAP.ASN
Display Name: DB2 DB4 SAMPLE QAPPLY ASN
Image Path:   ASNSERV DB2.DB4.SAMPLE.APP.AQ1 -ASNQCAP QCAPTURE_SERVER=SAMPLE CA
              PTURE_SCHEMA=ASN QCAPTURE_PATH=C:\PROGRA~1\SQLLIB
Dependency:   DB4-0
```

asntdiff: Comparing data in source and target tables

Use the **asntdiff** command to compare a source table with a target table and generate a list of differences between the two. Run the **asntdiff** command on Linux, UNIX, Windows, or z/OS at an operating system prompt or in a shell script.

The **asntdiff** command compares DB2 UDB tables on Linux, UNIX, Windows, z/OS, and iSeries operating systems.

Syntax



Parameters

Table 37 defines the invocation parameters for the **asntdiff** command.

Table 37. asntdiff invocation parameter definitions for Linux, UNIX, and Windows operating systems

Parameter	Definition
db=*server*	Specifies the DB2 UDB alias of the database that stores information about the source and target tables that will be compared. The value differs depending on whether you are using Q replication or SQL replication: **Q replication** The value is the name of the Q Capture server, which contains the IBMQREP_SUBS table. **SQL replication** The value is the name of the Apply control server, which contains the IBMSNAP_SUBS_MEMBR table.
schema=*schema*	Specifies the schema of the Q Capture control tables for Q replication, or the Apply control tables for SQL replication.
where=*WHERE_clause*	Specifies a SQL WHERE clause that uniquely identifies one row of the control table that stores information about the source and target tables that will be compared. The WHERE clause must be in double quotation marks. The value of this parameter differs depending on whether you are using Q replication or SQL replication: **Q replication** The WHERE clause specifies a row in the IBMQREP_SUBS table, using the SUBNAME column to identify the Q subscription that contains the source and target tables. **SQL replication** The WHERE clause specifies a row in the IBMSNAP_SUBS_MEMBR table, using the SET_NAME, APPLY_QUAL, TARGET_SCHEMA, and TARGET_TABLE columns to identify the subscription set member that contains the source and target tables.
diff=*table_name*	Specifies the name of the table that will be created in the source database to store differences between the source and target tables. The table will have one row for each difference that is detected. If you do not include this parameter, the difference table will be named *schema*.ASNTDIFF, where *schema* is the schema of the Q Capture control tables or Apply control tables that contain information about the source and target tables that you are comparing.

Examples for asntdiff

The following examples illustrate how to use the **asntdiff** command.

Example 1

In Q replication, to find the differences between a source and target table that are specified in a Q subscription named my_qsub, on a Q Capture server named source_db with a Q Capture schema of asn:

```
asntdiff db=source_db schema=asn where="where subname = 'my_qsub'"
```

Example 2

In SQL replication, to find the differences between a source and target table that are specified in a subscription set called my_set, with a target table named trg_table, on an Apply control server named apply_db, with an Apply schema of asn, and to name the difference table diff_table:

```
asntdiff DB=apply_db schema=asn where="where set_name = 'my_set'
 and target_table = 'trg_table'" diff=diff_table
```

Related concepts:

- "System commands for Q replication and event publishing—Overview" in the *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference*

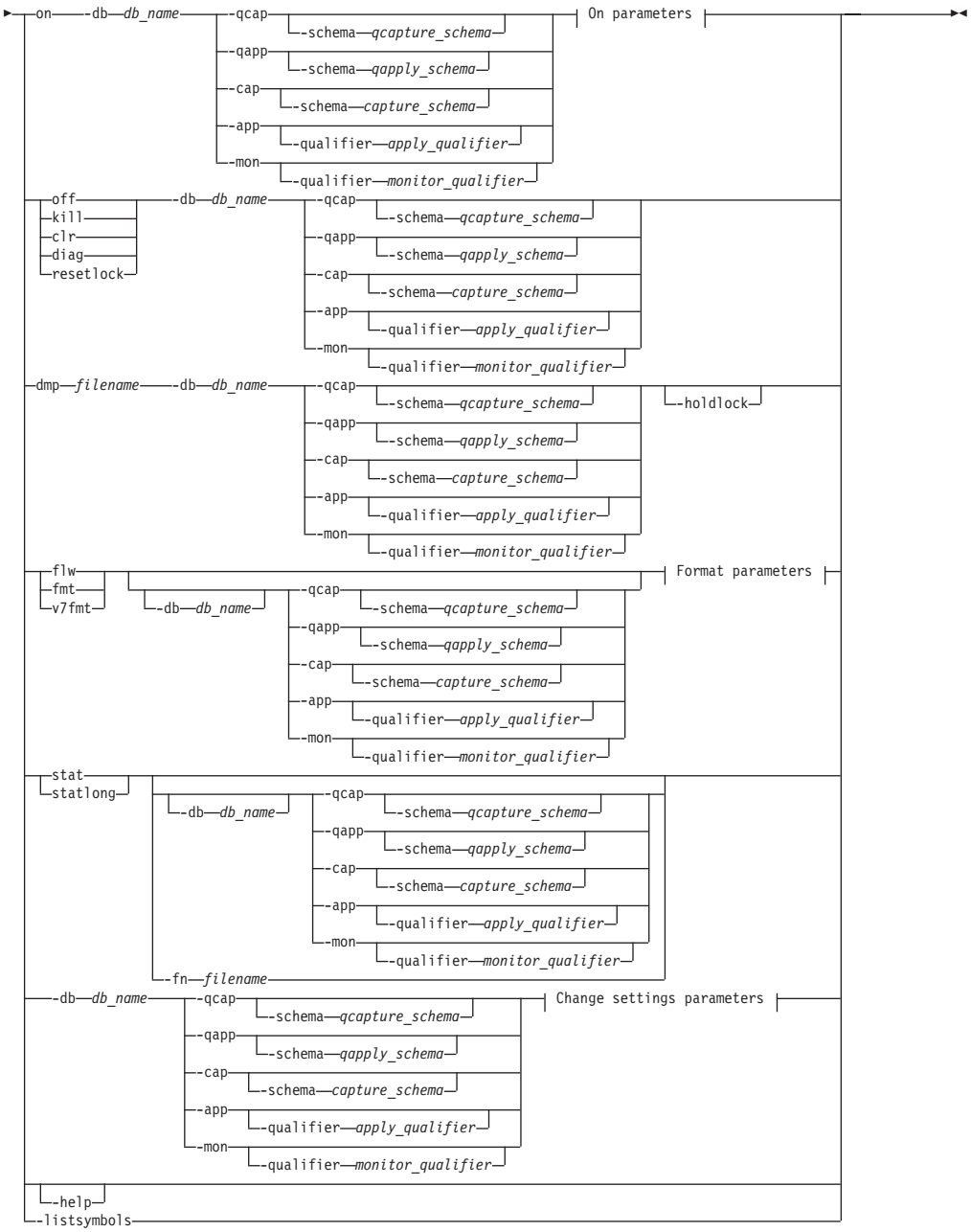
Related reference:

- "Road map: Q replication and event publishing system commands" in the *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference*
- "asntrep: Repairing differences between source and target tables" on page 316

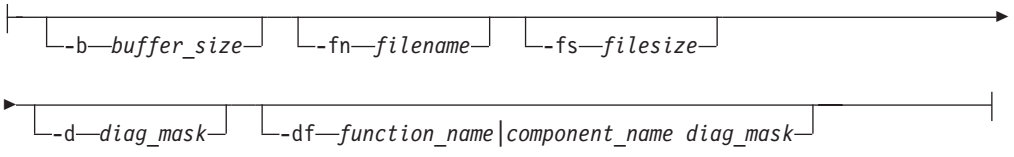
asntrc: Operating the replication trace facility

Use the **asntrc** command to run the trace facility on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. The trace facility logs program flow information from Q Capture, Q Apply, Capture, Apply, and Replication Alert Monitor programs. You can provide this trace information to IBM Software Support for troubleshooting assistance. Run this command at an operating system prompt or in a shell script.

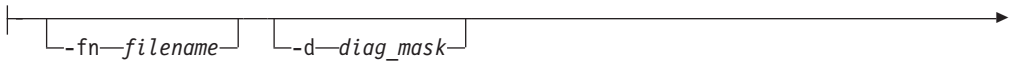
Syntax

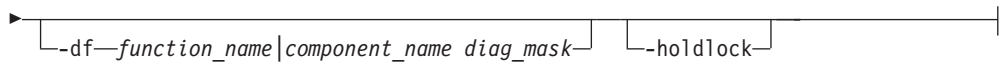


On parameters:

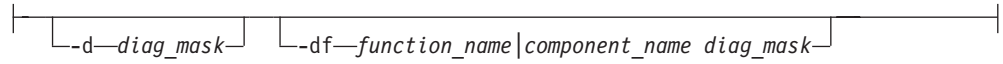


Format parameters:





Change settings parameters:



Parameters

Table 38 defines the invocation parameters for the **asntrc** command.

Table 38. asntrc invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
on	Specify to turn on the trace facility for a specific Q Capture, Q Apply, Capture, Apply, or Replication Alert Monitor program. The trace facility creates a shared memory segment used during the tracing process.
-db *db_name*	Specifies the name of the database to be traced: • Specifies the name of the Q Capture server for the Q Capture program to be traced. • Specifies the name of the Q Apply server for the Q Apply program to be traced. • Specifies the name of the Capture control server for the Capture program to be traced. • Specifies the name of the Apply control server for the Apply program to be traced. • Specifies the name of the Monitor control server for the Replication Alert Monitor program to be traced.
-qcap	Specifies that a Q Capture program is to be traced. The Q Capture program is identified by the **-schema** parameter.
-schema *qcapture_schema*	Specifies the name of the Q Capture program to be traced. The Q Capture program is identified by the Q Capture schema that you enter. Use this parameter with the **-qcap** parameter.
-qapp	Specifies that a Q Apply program is to be traced. The Q Apply program is identified by the **-schema** parameter.
-schema *qapply_schema*	Specifies the name of the Q Apply program to be traced. The Q Apply program is identified by the Q Apply schema that you enter. Use this parameter with the **-qapp** parameter.
-cap	Specifies that a Capture program is to be traced. The Capture program is identified by the **-schema** parameter.
-schema *capture_schema*	Specifies the name of the Capture program to be traced. The Capture program is identified by the Capture schema that you enter. Use this parameter with the **-cap** parameter.
-app	Specifies that an Apply program is to be traced. The Apply program is identified by the **-qualifier** parameter.

Table 38. asntrc invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
-qualifier *apply_qualifier*	Specifies the name of Apply program to be traced. This Apply program is identified by the Apply qualifier that you enter. Use this parameter with the **-app** parameter.
-mon	Specifies that a Replication Alert Monitor program is to be traced. The Replication Alert Monitor program is identified by the **-qualifier** parameter.
-qualifier *monitor_qualifier*	Specifies the name of Replication Alert Monitor program to be traced. This Replication Alert Monitor program is identified by the monitor qualifier that you enter. Use this parameter with the **-mon** parameter.
off	Specify to turn off the trace facility for a specific Q Capture, Q Apply, Capture, Apply, or Replication Alert Monitor program and free the shared memory segment in use.
kill	Specify to force an abnormal termination of the trace facility. Use this parameter only if you encounter a problem and are unable to turn the trace facility off with the **off** parameter.
clr	Specify to clear a trace buffer. This parameter erases the contents of the trace buffer but leaves the buffer active.
diag	Specify to view the filter settings while the trace facility is running.
resetlock	Specify to release the buffer latch of a trace facility. This parameter enables the buffer latch to recover from an error condition in which the trace program terminated while holding the buffer latch.
dmp *filename*	Specify to write the current contents of the trace buffer to a file.
-holdlock	Specifies that the trace facility can complete a file dump or output command while holding a lock, even if the trace facility finds insufficient memory to copy the buffer.
flw	Specify to display summary information produced by the trace facility and stored in shared memory or in a file. This information includes the program flow and is displayed with indentations that show the function and call stack structures for each process and thread.
fmt	Specify to display detailed information produced by the trace facility and stored in shared memory or in a file. This parameter displays the entire contents of the traced data structures in chronological order.
v7fmt	Specify to display information produced by the trace facility and stored in shared memory or in a file. This trace information appears in Version 7 format.
stat	Specify to display the status of a trace facility. This status information includes the trace version, application version, number of entries, buffer size, amount of buffer used, status code, and program timestamp.

Table 38. asntrc invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
statlong	Specify to display the status of a trace facility with additional z/OS version level information. This additional information includes the service levels of each module in the application and appears as long strings of text.
-fn *filename*	Specifies the file name containing the mirrored trace information, which includes all the output from the trace facility.
-help	Displays the valid command parameters with descriptions.
-listsymbols	Displays the valid function and component identifiers to use with the **-df** parameter.
-b *buffer_size*	Specifies the size of the trace buffer (in bytes). You can enter a `K` or an `M` after the number to indicate kilobytes or megabytes, respectively; these letters are not case sensitive.
-fs *filesize*	Specifies the size limit (in bytes) of the mirrored trace information file.
-d *diag_mask*	Specifies the types of trace records to be recorded by the trace facility. Trace records are categorized by a diagnostic mask number: **1** Flow data, which includes the entry and exit points of functions. **2** Basic data, which includes all major events encountered by the trace facility. **3** Detailed data, which includes the major events with descriptions. **4** Performance data. **Important:** The higher diagnostic mask numbers are *not* inclusive of the lower diagnostic mask numbers. You can enter one or more of these numbers to construct a diagnostic mask that includes only the trace records that you need. For example, specify **-d 4** to record only performance data; specify **-d 1,4** to record only flow and performance data; specify **-d 1,2,3,4** (the default) to record all trace records. Separate the numbers with commas. Enter a diagnostic mask number of 0 (zero) to specify that no global trace records are to be recorded by the trace facility. Type **-d 0** to reset the diagnostic level before specifying new diagnostic mask numbers for a tracing facility.
-df *function_name*|*component_name* *diag_mask*	Specifies that a particular function or component identifier is to be traced. Type the diagnostic mask number (1,2,3,4) after the function or component identifier name. You can enter one or more of these numbers. Separate the numbers with commas.

Examples for asntrc

The following examples illustrate how to use the **asntrc** command. These examples can be run on Linux, UNIX, Windows, or z/OS operating systems.

Example 1

To trace a running Capture program:

1. Start the trace facility, specifying a trace file name with a maximum buffer and file size:

   ```
   asntrc on -db mydb -cap -schema myschema -b 256k -fn myfile.trc -fs 500m
   ```

2. Start the Capture program, and let it run for an appropriate length of time.
3. While the trace facility is on, display the data directly from shared memory.

   To display the summary process and thread information from the trace facility:

   ```
   asntrc flw -db mydb -cap -schema myschema
   ```

   To view the flow, basic, detailed, and performance data records only from the Capture log reader:

   ```
   asntrc fmt -db mydb -cap -schema myschema -d 0
     -df "Capture Log Read" 1,2,3,4
   ```

4. Stop the trace facility:

   ```
   asntrc off -db mydb -cap -schema myschema
   ```

   The trace file contains all of the Capture program trace data that was generated from the start of the Capture program until the trace facility was turned off.
5. After you stop the trace facility, format the data from the generated binary file:

   ```
   asntrc flw -fn myfile.trc
   ```

   and

   ```
   asntrc fmt -fn myfile.trc -d 0 -df "Capture Log Read" 1,2,3,4
   ```

Example 2

To start a trace facility of a Replication Alert Monitor program:

```
asntrc on -db mydb -mon -qualifier monq
```

Example 3

To trace only performance data of an Apply program:

```
asntrc on -db mydb -app -qualifier aq1 -b 256k -fn myfile.trc -d 4
```

Example 4

To trace all flow and performance data of a Capture program:

```
asntrc on dbserv1 -cap -schema myschema -b 256k
  -fn myfile.trc -d 1,4
```

Example 5

To trace all global performance data and the specific Capture log reader flow data of a Capture program:

```
asntrc on -db mydb -cap -schema myschema -b 256k -fn myfile.trc -d 4
  -df "Capture Log Read" 1
```

Example 6

To trace a running Capture program and then display and save a point-in-time image of the trace facility:

1. Start the trace command, specifying a buffer size large enough to hold the latest records:

   ```
   asntrc on -db mydb -cap -schema myschema -b 4m
   ```

2. Start the Capture program, and let it run for an appropriate length of time.
3. View the detailed point-in-time trace information that is stored in shared memory:

   ```
   asntrc fmt -db mydb -cap -schema myschema
   ```

4. Save the point-in-time trace information to a file:

   ```
   asntrc dmp myfile.trc -db mydb -cap -schema myschema
   ```

5. Stop the trace facility:

   ```
   asntrc off -db mydb -cap -schema myschema
   ```

Examples for asntrc using shared segments

The standalone trace facility, **asntrc**, uses a shared segment to communicate with the respective Q Capture, Q Apply, Capture, Apply or Replication Alert Monitor programs to be traced. The shared segment will also be used to hold the trace entries if a file is not specified. Otherwise, matching options must be specified for both the **asntrc** command and for the respective programs to be traced to match the correct shared segment to control traces. The following examples show the options that need to be specified when the trace facility is used in conjunction with Q Capture, Q Apply, Capture, Apply or Alert Monitor programs.

With the Q Capture program, the database specified by the **-db** parameter with the **asntrc** command needs to match the database specified by the **capture_server** parameter with the **asnqcap** command:

```
asntrc -db ASN6 -schema EMI -qcap
asnqcap capture_server=ASN6 capture_schema=EMI
```

With the Q Apply program, the database specified by the **-db** parameter with the **asntrc** command needs to match the database specified by the **apply_server** parameter with the **asnqapp** command:

```
asntrc -db TSN3 -schema ELB -qapp
asnqapp apply_server=TSN3 apply_schema=ELB
```

With the Capture program, the database specified by the **-db** parameter with the **asntrc** command needs to match the database specified by the **capture_server** parameter with the **asncap** command:

```
asntrc -db DSN6 -schema JAY -cap
asncap capture_server=DSN6 capture_schema=JAY
```

With the Apply program, the database specified by the **-db** parameter with the **asntrc** command needs to match the database specified by the **control_server** parameter with the **asnapply** command:

```
asntrc -db SVL_LAB_DSN6 -qualifier MYQUAL -app
asnapply control_server=SVL_LAB_DSN6 apply_qual=MYQUAL
```

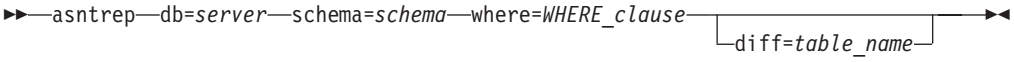
With the Replication Alert Monitor program, the database specified by the **-db** parameter with the **asntrc** command needs to match the database specified by the **monitor_server** parameter with the **asnmon** command:

```
asntrc -db DSN6 -qualifier MONQUAL -mon
asnmon monitor_server=DSN6 monitor_qual=MONQUAL
```

asntrep: Repairing differences between source and target tables

Use the **asntrep** command to synchronize a source and target table by repairing differences between the two tables. Run the **asntrep** command on Linux, UNIX, or Windows at an operating system prompt or in a shell script.

Syntax



Parameters

Table 39 defines the invocation parameters for the **asntrep** command.

Table 39. asntrep invocation parameter definitions for Linux, UNIX, and Windows operating systems

Parameter	Definition
db=*server*	Specifies the DB2 UDB alias of the database that stores information about the source and target tables that you want to synchronize. The value differs depending on whether you are using Q replication or SQL replication: **Q replication** The value is the name of the Q Capture server, which contains the IBMQREP_SUBS table. **SQL replication** The value is the name of the Apply control server, which contains the IBMSNAP_SUBS_MEMBR table.
schema=*schema*	Specifies the schema of the Q Capture control tables for Q replication, or the Apply control tables for SQL replication.

Table 39. asntrep invocation parameter definitions for Linux, UNIX, and Windows operating systems (continued)

Parameter	Definition
where=*WHERE_clause*	Specifies a SQL WHERE clause that uniquely identifies one row of the control table that stores information about the source and target tables that you are synchronizing. The WHERE clause must be in double quotation marks. The value of this parameter differs depending on whether you are using Q replication or SQL replication: **Q replication** The WHERE clause specifies a row in the IBMQREP_SUBS table, using the SUBNAME column to identify the Q subscription that contains the source and target tables. **SQL replication** The WHERE clause specifies a row in the IBMSNAP_SUBS_MEMBR table, using the SET_NAME, APPLY_QUAL, TARGET_SCHEMA, and TARGET_TABLE columns to identify the subscription set member that contains the source and target tables.
diff=*table_name*	Specifies the name of the table that was created in the source database using the **asntdiff** command to store differences between the source and target tables. The information that is stored in this table will be used to synchronize the source and target tables.

Examples for asntrep

The following examples illustrate how to use the **asntrep** command.

Example 1

In Q replication, to synchronize a source and target table that are specified in a Q subscription named my_qsub, on a Q Capture server named source_db, with a Q Capture schema of asn, and whose differences are stored in a table called q_diff_table:

```
asntrep db=source_db schema=asn where="where subname = 'my_qsub'" diff=q_diff_table
```

Example 2

In SQL replication, to synchronize a source and target table that are specified in a subscription set called my_set, with a target table named trg_table, on an Apply control server named apply_db, with an Apply schema of asn, and whose differences are stored in a table called sql_diff_table:

```
asntrep DB=apply_db SCHEMA=asn WHERE="where set_name = 'my_set'
 and target_table = 'trg_table'" diff=sql_diff_table
```

Related concepts:

- "System commands for Q replication and event publishing—Overview" in the *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference*

Related reference:

- "Road map: Q replication and event publishing system commands" in the *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference*
- "asntdiff: Comparing data in source and target tables" on page 307

Chapter 19. System commands for SQL replication (OS/400)

This chapter describes the replication commands that run under the OS/400 operating system on iSeries servers. You can enter these commands at an operating system command prompt or through a command line program.

This chapter contains a section for each command. Each section contains a brief description of the command, a syntax diagram, and a table of parameters with corresponding definitions. The end of each section has examples of command usage and cross-references to related information.

The commands include:

- "ADDDPRREG: Adding a DPR registration (OS/400)"
- "ADDDPRSUB: Adding a DPR subscription set (OS/400)" on page 327
- "ADDDPRSUBM: Adding a DPR subscription-set member (OS/400)" on page 343
- "ANZDPR: Operating the Analyzer (OS/400)" on page 352
- "CHGDPRCAPA: Changing DPR Capture attributes (OS/400)" on page 355
- "CRTDPRTBL: Creating the replication control tables (OS/400)" on page 360
- "ENDDPRAPY: Stopping Apply (OS/400)" on page 361
- "ENDDPRCAP: Stopping Capture (OS/400)" on page 364
- "GRTDPRAUT: Authorizing users (OS/400)" on page 366
- "INZDPRCAP: Reinitializing DPR Capture (OS/400)" on page 374
- "OVRDPRCAPA: Overriding DPR capture attributes (OS/400)" on page 375
- "RMVDPRREG: Removing a DPR registration (OS/400)" on page 380
- "RMVDPRSUB: Removing a DPR subscription set (OS/400)" on page 381
- "RMVDPRSUBM: Removing a DPR subscription-set member (OS/400)" on page 383
- "RVKDPRAUT: Revoking authority (OS/400)" on page 384
- "STRDPRAPY: Starting Apply (OS/400)" on page 386
- "STRDPRCAP: Starting Capture (OS/400)" on page 393
- "WRKDPRTRC: Using the DPR trace facility (OS/400)" on page 400

ADDDPRREG: Adding a DPR registration (OS/400)

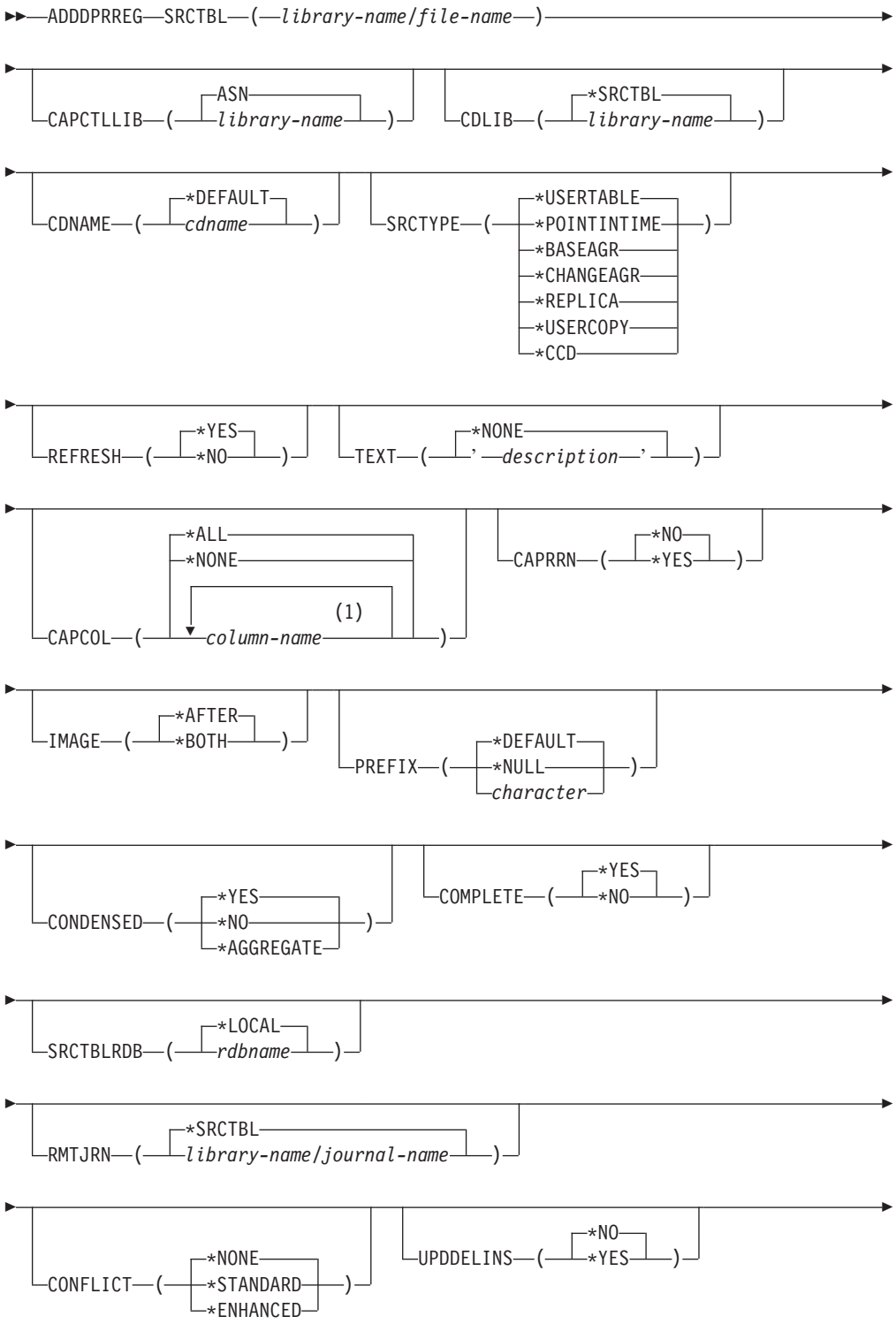
Use the Add DPR registration (**ADDDPRREG**) command to register a table as a source table for DB2 DataPropagator for iSeries.

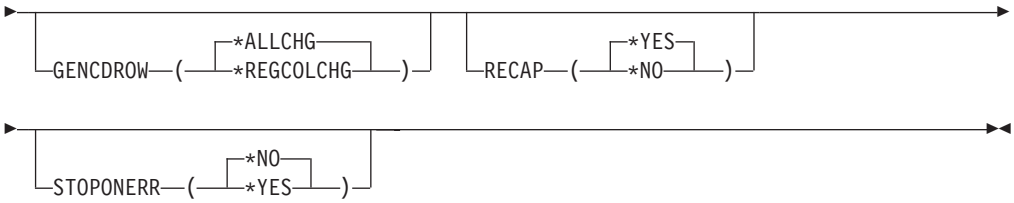
Restriction: You can register a table only if the ASN (Capture schema) library is in the same Auxiliary Pool (either base or independent ASP) where the ASN library is located.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To register a table using the ADDDPRREG command:





Notes:

1 You can specify up to 300 column names.

Table 40 lists the invocation parameters.

Table 40. ADDDPRREG command parameter definitions for OS/400

Parameter	Definition and prompts
SRCTBL	Specifies the table that you want to register as a source table. The Capture program supports any physical file in an OS/400 library or collection that is externally defined and in single format. This parameter is required. *library-name/file-name* Represents the qualified name of the table that you want to register.
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. **ASN** (default) The Capture control tables reside in the ASN library. *library-name* The name of the library that contains the Capture control tables. You can create this library using the **CRTDPRTBL** command with the **CAPCTLLIB** parameter.
CDLIB	Specifies the library in which the change-data (CD) table for this registered source is created. ***SRCTBL** (default) Creates the CD table in the library in which the source table resides. *library-name* Creates the CD table in this specified library name.
CDNAME	Specifies the name of the change-data (CD) table. ***DEFAULT** (default) Creates the CD table with the default name, which is based on the current timestamp. For example, if the current timestamp is January 23, 2002 at 09:58:26, the default name is ASN020123095826CD. *cdname* Creates the CD table with this specified name.

Table 40. ADDDPRREG command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
SRCTYPE	Specifies the type of source table that you are registering. Choose a source type based on your replication configuration: • Use the default of USERTABLE for a basic data distribution or a data consolidation configuration. • Use REPLICA for an update-anywhere configuration. • Use POINTINTIME, BASEAGR, CHANGEAGR, USERCOPY, or CCD if you have a multi-tier configuration and want the target table to be a source for a subsequent tier in your replication configuration. If you are registering an existing target table as a source, the registration fails if the target table does not contain the IBMSNAP table columns indicated by the specified source type.
SRCTYPE (continued)	***USERTABLE** (default) A user database table, which is the most common type of registered table. The table cannot contain any columns that start with a DB2 DataPropagator for iSeries column identifier of either IBMSNAP or IBMQSQ. ***POINTINTIME** A point-in-time copy table, which includes content that matches all or part of the content of a source table and a DB2 DataPropagator for iSeries system column that identifies the time when a particular row was last inserted or updated at the source system. The table must contain the IBMSNAP_LOGMARKER timestamp column and can optionally contain an INTEGER column called IBMQSQ_RRN. ***BASEAGR** A base aggregate copy, which contains data aggregated at intervals from a user table or from a point-in-time table. The base aggregate table must contain the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER timestamp columns. ***CHANGEAGR** A change aggregate copy table, which contains data aggregations that are based on changes recorded for a source table. The table must contain the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER timestamp columns. ***REPLICA** A target table for a replica subscription. Register this type of table so that changes from the target table are replicated back to the original source table. This table cannot contain any DB2 DataPropagator for iSeries system columns or any columns that start with the DB2 DataPropagator for iSeries column identifier of either IBMSNAP or IBMQSQ. The table contains all of the columns from the original source table. ***USERCOPY** A target table with content that matches all or part of the content of a source table. The user copy table contains only user data columns.

Table 40. ADDDPRREG command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
SRCTYPE (continued)	***CCD** A consistent-change data (CCD) table, which contains transaction-consistent data from the source table. The table must contain columns that are defined as follows: • IBMSNAP_INTENTSEQ CHAR(10) FOR BIT DATA NOT NULL • IBMSNAP_OPERATION CHAR(1) NOT NULL • IBMSNAP_COMMITSEQ CHAR(10) FOR BIT DATA NOT NULL • IBMSNAP_LOGMARKER TIMESTAMP NOT NULL
REFRESH	Specifies whether the full-refresh capability is enabled. You can use this value to turn off the capability of the Apply program to perform a full refresh from the source database. ***YES** (default) Full refreshes are allowed. ***NO** Full refreshes are not allowed. If the target table is a base aggregate or change aggregate, you should set this parameter to *NO.
TEXT	Specifies the textual description that is associated with this registration. ***NONE** (default) No description is associated with the entry. *description* The textual description of this registration. You can enter a maximum of 50 characters and must enclose the text in single quotation marks.
CAPCOL	Specifies the columns for which changes are captured for this registered table. ***ALL** (default) Changes are captured for all columns. ***NONE** Changes are not captured for this table. Use this value to specify that you want this table registered for full refresh only. The change-data (CD) table is not created with this registered table, and the Capture program will not capture changes for the table. *column-name* The column names for which changes are captured. You can type up to 300 column names. Separate the column names with spaces.
CAPRRN	Specifies whether the relative record number (RRN) of each changed record is captured. ***NO** (default) The relative record number is not captured. ***YES** The relative record number is captured. An additional column called IBMQSQ_RRN is created in the change-data (CD) table. Set this parameter to *YES only if there are no unique keys in the source table.

Table 40. ADDDPRREG command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
IMAGE	Specifies whether the change-data (CD) table contains both before and after images of the changes to the source table. This applies globally to all columns specified on the Capture columns (**CAPCOL**) parameter. This **IMAGE** parameter is not valid when the **CAPCOL** parameter is set to *NONE. The source table must be journaled with *BOTH images even if you specify *AFTER on this parameter. ***AFTER** (default) The Capture program records only after images of the source table in the CD table. ***BOTH** The Capture program records both before images and after images of the source table in the CD table.
PREFIX	Specifies the prefix character identifying before-image column names in the change-data (CD) table. You must ensure that none of the registered column names of the source table begins with this prefix character. ***DEFAULT** (default) The default prefix (@) is used. ***NULL** No before images are captured. This value is not valid if the **IMAGE** parameter is set to *BOTH. *character* Any single alphabetic character that is valid in an object name.
CONDENSED	Specifies whether the source table is condensed. A condensed table contains current data with no more than one row for each primary key value in the table. ***YES** (default) The source table is condensed. ***NO** The source table is not condensed. ***AGGREGATE** The source table type is either *BASEAGR (base aggregate) or *CHANGEAGR (change aggregate). If this value is used, you must set the **COMPLETE** parameter to *NO.
COMPLETE	Specifies whether the source table is complete, which means that the table contains a row for every primary key value of interest. ***YES** (default) The source table is complete. ***NO** The source table is not complete.

Table 40. ADDDPRREG command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
SRCTBLRDB	Specifies whether you want to use remote journaling, in which the source table and the remote journal reside on different systems. Use this parameter to specify the location of the source table. ***LOCAL** (default) The source table resides locally (on the machine where you are running the **ADDDPRREG** command). *rdbname* The name of the relational database where the source table exists. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this relational database name.
RMTJRN	Specifies the name of the remote journal when the name of this journal and the name of the journal on the source system are different. You must issue this command from the system where the remote journal resides. ***SRCTBL** (default) The remote journal name is the same as the journal name of the source table. *library-name/journal-name* The qualified library and journal name that reside on this system and are used for journaling the remote source table. You can specify a remote journal name only if you specified a remote source table location using the **SRCTBLRDB** parameter.
CONFLICT	Specifies the conflict level that is used by the Apply program when detecting conflicts in a replica subscription. ***NONE** (default) No conflict detection. ***STANDARD** Moderate conflict detection. The Apply program searches for conflicts in rows that are already captured in the replica change-data (CD) tables. ***ENHANCED** Enhanced conflict detection. This option provides the best data integrity among all replicas and source tables.
UPDDELINS	Determines how the Capture program stores updated source data in the change-data (CD) table. ***NO** (default) The Capture program stores each source change in a single row in the CD table. ***YES** The Capture program stores each source change using two rows in the CD table, one for the delete and one for the insert. The Apply program processes the delete row first and the insert row second.

Table 40. ADDDPRREG command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
GENCDROW	Specifies whether the Capture program captures changes from all rows in the source table. ***ALLCHG** (default) The Capture program captures changes from all rows in the source table (including changes in unregistered columns) and adds these changes to the change-data (CD) table. ***REGCOLCHG** The Capture program captures changes only if the changes occur in registered columns. The Capture program then adds these rows to the CD table. You cannot specify *REGCOLCHG if the **CAPCOL** parameter is set to *ALL or *NONE.
RECAP	Specifies whether the changes made by the Apply program are recaptured by the Capture program. ***YES** (default) Changes made to the source table by the Apply program are captured and entered into the change-data (CD) table. ***NO** Changes that were made to the source table by the Apply program are not captured and, therefore, do not appear in the CD table. You should use this option when registering REPLICA type tables.
STOPONERR	Specifies whether the Capture program stops when it encounters an error.[1] ***NO** (default) The Capture program does not stop when it encounter an error. The Capture program issues messages, deactivates the registration that caused the error, and then continues processing. ***YES** The Capture program issues messages and then stops when it encounters an error.

Notes:

1. If this parameter is set to Yes (Y), the Capture journal job stops while other journal jobs continue to run. If this parameter is set to No (N), the Capture program stops the registration file that contains the error.

   This parameter also sets the columns in the register table rows. The STATE column is set to 'S' and the STATE_INFO column to is set 200A*xxxx* where *xxxx* is the reason code. To set the registration back to the Action ('A') state, perform the following steps:

   - Correct the ASN200A message. Refer to the appropriate OS/400 documentation for the corrected action.
   - Use the Replication Center or the OS/400 command STRSQL to set the columns in the IBMSNAP_REGISTER table row. Set the STATE column to 'A', and the STATE_INFO column to null.
   - If Capture is running, issue the INZDPRCAP command to reinitialize data replication for that journal.

Examples for ADDDPRREG

The following examples illustrate how to use the **ADDDPRREG** command.

Example 1

To register a source table named EMPLOYEE from the HR library under the default Capture schema:

```
ADDDPRREG SRCTBL(HR/EMPLOYEE)
```

Example 2

To register a source table named EMPLOYEE from the HR library under the BSN Capture schema and to create a CD table named CDEMPLOYEE under the HRCDLIB library:

```
ADDDPRREG SRCTBL(HR/EMPLOYEE) CAPCTLLIB(BSN) CDLIB(HRCDLIB) CDNAME(CDEMPLOYEE)
```

Example 3

To register a source table with a source type of point-in-time that is named SALES from the DEPT library under the BSN Capture schema:

```
ADDDPRREG SRCTBL(DEPT/SALES) CAPCTLLIB(BSN) SRCTYPE(*POINTINTIME)
```

Example 4

To register a source table named SALES from the DEPT library and to specify that the CD table contains both before and after images of source table changes:

```
ADDDPRREG SRCTBL(DEPT/SALES) IMAGE(*BOTH)
```

Example 5

To register a source table named SALES from the DEPT library of the relational database named RMTRDB1 using remote journals:

```
ADDDPRREG SRCTBL(DEPT/SALES) SRCTBLRDB(RMTRDB1) RMTJRN(RMTJRNLIB/RMTJRN)
```

Example 6

To register the EMPLOYEE source table from the HR library and to capture changes only for the EMPNO, NAME, DEPT, and NETPAY columns:

```
ADDDPRREG SRCTBL(HR/EMPLOYEE) CAPCOL(EMPNO NAME DEPT NETPAY)
```

Related tasks:

• Chapter 3, “Registering tables and views as SQL replication sources,” on page 35

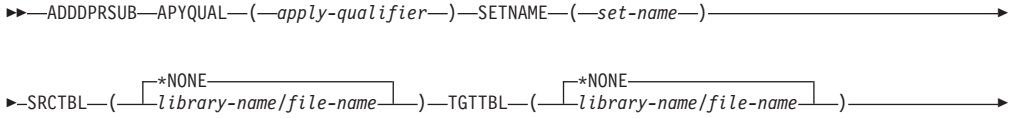
ADDDPRSUB: Adding a DPR subscription set (OS/400)

Use the Add DPR subscription set (**ADDDPRSUB**) command to create a subscription set with either one member or no members.

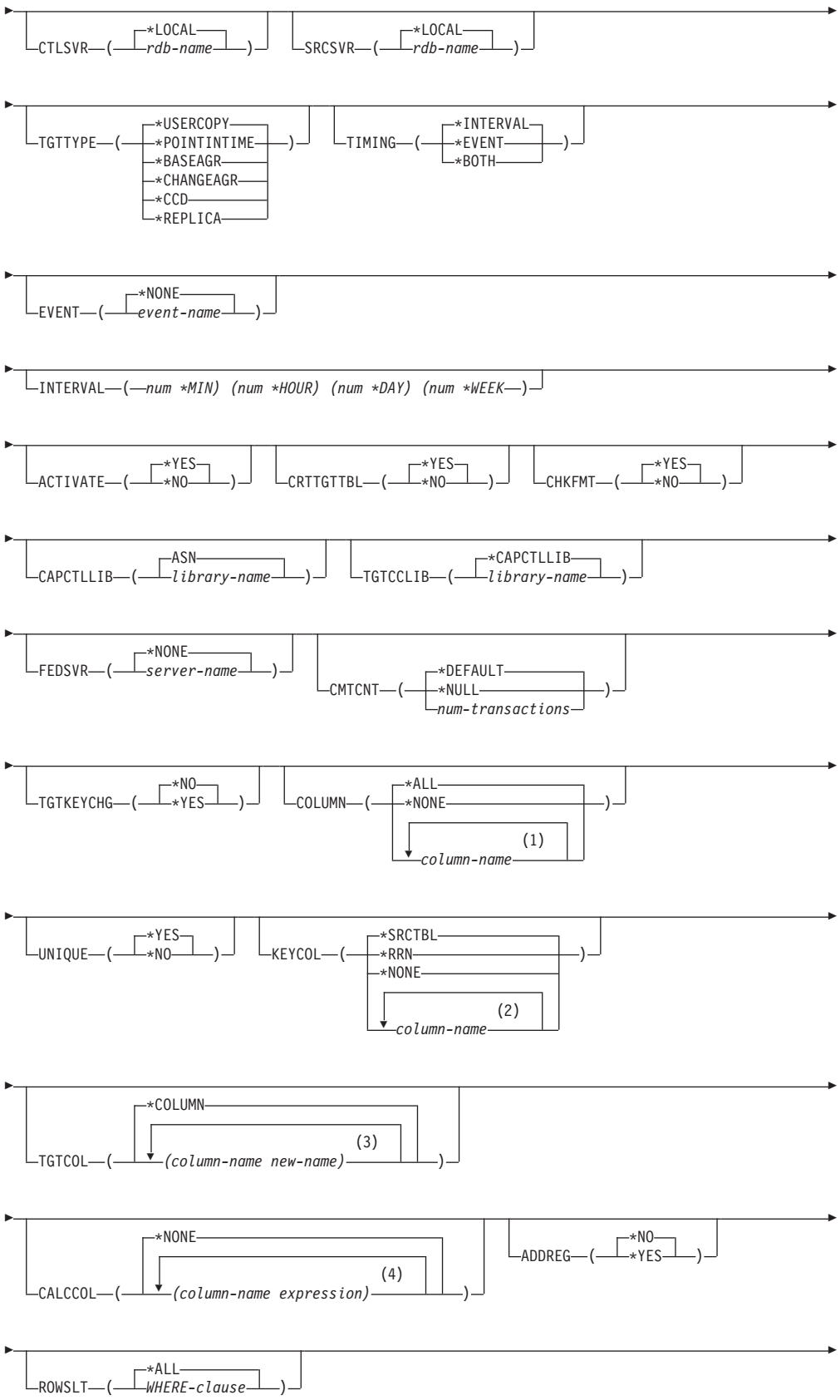
After you type the command name on the command line, you can press the F4 key to display the command syntax.

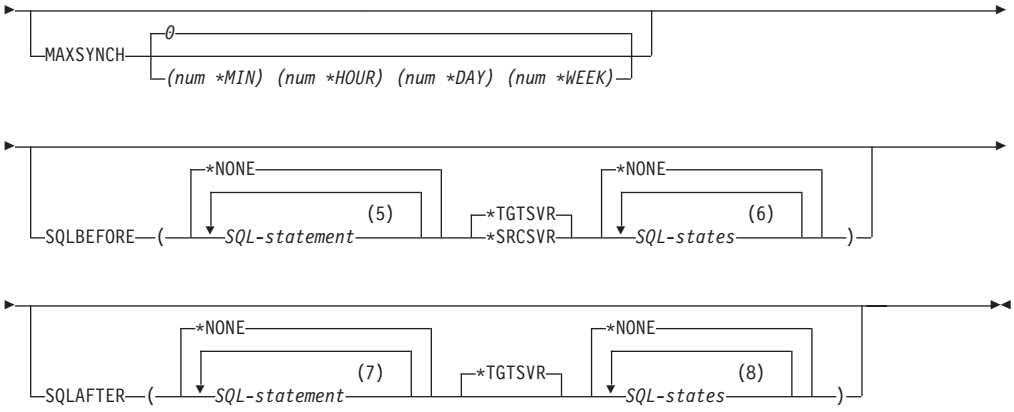
To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To create a subscription set using the ADDDPRSUB command:



ADDDPRSUB


CTLSVR
*LOCAL
rdb-name
SRCSVR
*LOCAL
rdb-name
TGTTYPE
*USERCOPY
*POINTINTIME
*BASEAGR
*CHANGEAGR
*CCD
*REPLICA
TIMING
*INTERVAL
*EVENT
*BOTH
EVENT
*NONE
event-name
INTERVAL
num *MIN) (num *HOUR) (num *DAY) (num *WEEK
ACTIVATE
*YES
*NO
CRTTGTTBL
*YES
*NO
CHKFMT
*YES
*NO
CAPCTLLIB
ASN
library-name
TGTCCLIB
*CAPCTLLIB
library-name
FEDSVR
*NONE
server-name
CMTCNT
*DEFAULT
*NULL
num-transactions
TGTKEYCHG
*NO
*YES
COLUMN
*ALL
*NONE
(1)
column-name
UNIQUE
*YES
*NO
KEYCOL
*SRCTBL
*RRN
*NONE
(2)
column-name
TGTCOL
*COLUMN
(3)
(column-name new-name)
CALCCOL
*NONE
(4)
(column-name expression)
ADDREG
*NO
*YES
ROWSLT
*ALL
WHERE-clause



Notes:

1 You can specify up to 300 column names.

2 You can specify up to 120 column names.

3 You can specify up to 300 column names.

4 You can specify up to 100 column names and expressions.

5 You can specify up to 3 SQL statements.

6 You can specify up to 10 SQLSTATES.

7 You can specify up to 3 SQL statements.

8 You can specify up to 10 SQLSTATES.

Table 41 lists the invocation parameters.

Table 41. ADDDPRSUB command parameter definitions for OS/400

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier that identifies which Apply program processes this subscription set. Subscription sets under an Apply qualifier run in a separate job. This parameter is required. *apply-qualifier* The name of the Apply qualifier.
SETNAME	Specifies the subscription-set name. This parameter is required. *set-name* The name of the subscription set. The subscription-set name that you enter must be unique for the specified Apply qualifier or the **ADDDPRSUB** command produces an error. Because the Apply program handles the set of target tables as a group, when one target table fails for any reason, the entire subscription set fails.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
SRCTBL	Specifies the name of the source table that is used to copy information into your subscription set. You must register this table at the Capture control server before this table can become a member of a subscription set. This parameter is required. ***NONE** (default) This subscription set does not have a source member. Use when creating a subscription set without members. *library-name/file-name* The qualified name of the source table. Use when creating a subscription set with one member.
TGTTBL	Specifies the name of the target table. The target table is automatically created if you set the **CRTTGTTBL** parameter to *YES and the target table does not already exist. This parameter is required. ***NONE** (default) This subscription set does not have a target member. Use when creating a subscription set without members. *library-name/file-name* The qualified name of the target table. Use when creating a subscription set with one member.
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL** (default) The Apply control tables reside locally (on the machine from which you are running the **ADDDPRSUB** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.
SRCSVR	Specifies the relational database name of the system that contains the Capture control tables. ***LOCAL** (default) The source table is registered on the local machine (the machine from which you are running the **ADDDPRSUB** command). *rdb-name* The name of the relational database where the Capture control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TGTTYPE	Specifies the target table type. After you create a target table as one of these types, you can use this parameter value on the **SRCTBL** parameter of the Add DPR Registration (**ADDDPRREG**) command to register this target table as a source table for multi-tier replication. ***USERCOPY** (default) The target table is a user copy, which is a target table with content that matches all or part of the content of a source table. A user copy is handled like a point-in-time copy but does not contain any of the DB2 DataPropagator for iSeries system columns that are present in the point-in-time target table. This value is not valid when a value of *RRN is specified on the **KEYCOL** parameter. The table that you specified with the **SRCTBL** parameter must be one of the following types: user database, point-in-time copy, or consistent-change data (CCD). **Important:** If the target table already exists, DB2 DataPropagator for iSeries does not automatically journal changes to it. You must start journaling outside of DB2 DataPropagator for iSeries.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TGTTYPE (continued)	***POINTINTIME** The target table is a point-in-time copy. A point-in-time copy is a target table with content that matches all or part of the content of the source table and includes the DB2 DataPropagator for iSeries system column (IBMSNAP_LOGMARKER), which identifies when a particular row was inserted or updated at the Capture control server. ***BASEAGR** The target table is a base aggregate copy, which is a target table that contains data that is aggregated (calculated) from a source table. The source table for a base aggregate target must be either a user table or a point-in-time table. This target table must contain the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER system timestamp columns. ***CHANGEAGR** The table is a change aggregate copy, which is a target table that contains data that is aggregated (calculated) based on the contents of a change-data (CD) table. This target table is created with the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER system timestamp columns. ***CCD** The table is a consistent-change data (CCD) table, which is a target table created from a join of data in the change-data (CD) table and the unit-of-work (UOW) table. A CCD table provides transaction-consistent data for the Apply program and must include the following columns: • IBMSNAP_INTENTSEQ • IBMSNAP_OPERATION • IBMSNAP_COMMITSEQ • IBMSNAP_LOGMARKER ***REPLICA** The target table is a replica table, which is used only for update-anywhere replication. The replica target table receives changes from the master source table, and changes to the replica target table are propagated back to the master source table. A replica table is automatically registered as a source table.
TIMING	Specifies the type of timing (scheduling) that the Apply program uses to process the subscription set. ***INTERVAL** (default) The Apply program processes the subscription set at a specific time interval (for example, once a day). ***EVENT** The Apply program processes the subscription set when a specific event occurs. ***BOTH** The Apply program processes the subscription set either at a specific interval or when an event occurs, whichever occurs first.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
EVENT	Specifies an event. The event that you enter must match an event name in the subscription events (IBMSNAP_SUBS_EVENT) table. ***NONE** (default) No event is used. *event-name* A unique character string that represents an event described in the IBMSNAP_SUBS_EVENT table.
INTERVAL	Specifies the time interval (weeks, days, hours, and minutes) from start time to start time between refreshes of the target copy. This is a two-part value. The first part is a number; the second part is the unit of time: ***MIN** Minutes ***HOUR** Hours ***DAY** Days ***WEEK** Weeks You can specify combinations of numbers with units of time. For example, ((2 *WEEK) (3 *DAY) (35 *MIN)) specifies a time interval of two weeks, three days, and 35 minutes. If you specify multiple occurrences of the same unit of time, the last occurrence is used.
ACTIVATE	Specifies whether the subscription set is active. The Apply program does not process this subscription set unless this parameter is set to *YES. ***YES** (default) The subscription set is active. ***NO** The subscription set is not active.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
CRTTGTTBL	Specifies whether the target table (or view) is created. ***YES** (default) Creates the target table (or view) if it does not exist. Otherwise, the existing table or view becomes the target, and the format of this existing table or view is checked if the value of the **CHKFMT** parameter is set to *YES. An additional index, with the values that you specified by the **UNIQUE** and **KEYCOL** parameters, is created for a target table if no such index currently exists. The command fails if an existing target table contains rows that violate the conditions of the additional index. ***NO** Does not create the target table or view. You must create the table or view with the correct attributes before starting the Apply program. If the table or view exists and you set **CHKFMT** to *YES, the **ADDDPRSUB** command ensures that the format of the existing table matches the subscription-set definition that you set. If **CHKFMT** is *NO, you must ensure that the format of the existing table matches the subscription-set definition. **Important:** If the table or view already exists, DB2 DataPropagator for iSeries does not automatically journal changes to the existing object. You must start journaling outside of DB2 DataPropagator for iSeries.
CHKFMT	Specifies whether DB2 DataPropagator for iSeries checks the subscription set and the target table to ensure that the columns match. This parameter is ignored if the **CRTTGTTBL** parameter is *YES; this parameter is also ignored if the **CRTTGTTBL** parameter is set to *NO and the target table does not exist. ***YES** (default) DB2 DataPropagator for iSeries verifies that the columns defined for this subscription set match the columns in the target table. This command fails if a mismatch is detected. ***NO** DB2 DataPropagator for iSeries ignores the differences between the subscription set and the existing target table. You must ensure that the target table is compatible with the subscription set.
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. These Capture control tables process the source for this subscription set. **ASN** (default) The Capture control tables reside in the ASN library. *library-name* The name of a library that contains the Capture control tables. This is the library in which the source table was registered.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TGTCCLIB	Specifies the target control library. ***CAPCTLLIB** (default) The target control library is the same library in which the Capture control tables reside. *library-name* The name of a library that contains the target control tables. If you are using a target table as a source for another subscription set (such as an external CCD table), this parameter value is the Capture schema when this table is used as a source.
FEDSVR	Specifies whether a federated database system is the source for this subscription set. ***NONE** (default) The source server is not a federated database system. *server-name* The name of the federated database system for this subscription set (for non-DB2 relational sources).
CMTCNT	Specifies the commitment count, which is the number of transactions that the Apply program processes before a commit. ***DEFAULT** (default) The command determines the value to use. If the **TGTTYPE** is set to *REPLICA, then the **CMTCNT** is zero (0). If the **TGTTYPE** is anything other than *REPLICA, the **CMTCNT** is null. ***NULL** The subscription set is read-only. The Apply program will fetch answer sets for the subscription-set members one member at a time, until all data has been processed and then will issue a single commit for the entire subscription set. *num-transactions* Specifies the number of transactions processed before the Apply program commits the changes. This parameter is valid only if the **TGTTYPE** parameter is set to *REPLICA.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TGTKEYCHG	Specifies how the Apply program handles updates when changes occur in source columns that are part of the target key columns for the target table. This parameter works in conjunction with the **USEDELINS** parameter on the **ADDDPRREG** command: • If **USEDELINS** is YES and **TGTKEYCHG** is YES, updates are not allowed. • If **USEDELINS** is YES and **TGTKEYCHG** is NO, updates become delete and insert pairs. • If **USEDELINS** is NO and **TGTKEYCHG** is YES, the Apply program handles this condition with special logic. • If **USEDELINS** is NO and **TGTKEYCHG** is NO, the Apply program processes the changes as normal updates. ***NO** (default) Updates to the source table are staged by the Capture program and processed by the Apply program to the target table. ***YES** The Apply program updates the target table based on the before images of the target key column, meaning that the Apply program changes the predicate to the old values instead of the new.
COLUMN	Specifies the columns to be included in the target table. The column names must be unqualified. Choose the column names from the list of column names that you specified with the **CAPCOL** parameter when you registered the source table. If you set the **IMAGE** parameter to *BOTH when registering this table, you can specify before-image column names. The before-image column names are the original column names with a prefix. This prefix is the character that you specified in the **PREFIX** parameter of the **ADDDPRREG** command. ***ALL** (default) All of the columns that you registered in the source are included in the target table. ***NONE** No columns from the source table are included in the target table. You can use *NONE when you want only computed columns in the target table. This value is required if the **CALCCOL** parameter contains summary functions but no GROUP BY is performed. *column-name* The names of up to 300 source columns that you want to include in the target table. Separate the column names with spaces.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
UNIQUE	Specifies whether the target table has unique keys as indicated by the **KEYCOL** parameter. ***YES** (default) The target table supports one net change per key; only one row exists in the target table for that key regardless of how many changes are made to the key. This value specifies that the table contains current data rather than a history of changes to the data. A condensed table includes no more than one row for each primary key value in the table and can be used to supply current information for a refresh. ***NO** The target table supports multiple changes per key. The changes are appended to the target table. This value specifies that the table contains a history of changes to the data rather than current data. A non-condensed table includes more than one row for each key value in the table and can be used to supply a history of changes to the data. A non-condensed table cannot supply current data for a refresh.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
KEYCOL	Specifies columns that describe the key of the target table. The column names must be unqualified. For *POINTINTIME, *REPLICA, and *USERCOPY target tables (as specified on the **TGTTYPE** parameter), you must identify one or more columns as the target key for the target table. This target key is used by the Apply program to identify each unique row that changes during change-capture replication. ***SRCTBL** (default) The key columns in the target table are the same as those in the source table. The **ADDDPRREG** command uses the key that is specified in the source table if the source table is keyed. The following key columns are used: • Key columns that you defined through DDS when creating the table with the Create Physical File (**CRTPF**) command • Primary and unique keys that you defined with the CREATE TABLE and ALTER TABLE SQL statements • Unique keys that you defined with the CREATE INDEX SQL statements If you use a column as a key more than once and with different ordering, the target table key is defined with ascending order. ***RRN** The key column in the target table is the IBMQSQ_RRN column. The target table is created with an IBMQSQ_RRN column, and this column is used as the key. When the Apply program runs, if the source table is a user table and the target table is a point-in-time or user copy, the IBMQSQ_RRN column in the target table is updated with the relative record number of the associated record in the source table. Otherwise, the IBMQSQ_RRN column in the target table is updated with the value of the IBMQSQ_RRN column in the source table. ***NONE** The target copy does not contain a target key. You cannot specify *NONE if the target table type is *POINTINTIME, *REPLICA, or *USERCOPY. *column-name* The names of the target columns that you want to use as the target key columns. You can specify up to 120 column names. Separate the column names with spaces.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TGTCOL	Specifies the new names for all the columns that the Apply program updates in the target table. These names override the column names taken from the source table. The column names must be unqualified. If you specified a value of *NONE for the **COLUMN** parameter, do not use this parameter. Use this parameter to give more meaningful names to the target table columns. Specify each source column name and the name of the corresponding column on the target table. ***COLUMN** (default) The target columns are the same as the columns you specified in the **COLUMN** parameter. *column-name* The column names from the source table that you want to change at the target. You can list up to 300 column names. *new-name* The new names of the target columns. You can list up to 300 new column names. If you do not use this parameter, the name of the column on the target table will be the same as the source column name.
CALCCOL	Specifies the list of user-defined or calculated columns in the target table. The column names must be unqualified. Enclose each column name and expression pair in parenthesis. You must specify a column name for each SQL expression. If you want to define any column as an SQL expression without a GROUP BY statement, you must set the **COLUMN** parameter to *NONE. ***NONE** (default) No user-defined or calculated columns are included in the target table. *column-name* The column names of the user-defined or calculated columns in the target table. You can list up to 100 column names. *expression* The expressions for the user-defined or calculated columns in the target table. You can list up to 100 SQL column expressions.
ADDREG	Specifies whether the target table is automatically registered as a source table. Use this parameter to register CCD target type tables. ***NO** (default) The target table is not registered as a source table. DB2 DataPropagator for iSeries ignores this parameter value if the target type is *REPLICA. Replica target tables are always automatically registered as source tables. ***YES** The target table is registered as a source table. This command fails if you already registered the target table. Do not set this parameter to *YES if the target table type is *USERCOPY, *POINTINTIME, *BASEAGR, or *CHANGEAGR. If you set the **CRTTGTTBL** parameter to *NO, you must create the target table before attempting to register it as a source.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
ROWSLT	Specifies the predicates to be placed in an SQL WHERE clause. The Apply program uses these predicates to determine which rows in the change-data (CD) table of the source to apply to the target table. Use this parameter if you want only a subset of the source changes to be replicated to the target table. ***ALL** (default) The Apply program applies all changes in the CD table to the target table. *WHERE-clause* The SQL WHERE clause that specifies which rows from the CD table the Apply program applies to the target table. Do not include the WHERE keyword; it is implied on this parameter. This WHERE clause must be valid on the data server you are using to run the clause. **Note:** The WHERE clause on this parameter is unrelated to any WHERE clauses specified on the SQLBEFORE or SQLAFTER parameters.
MAXSYNCH	Specifies the maximum synch minutes. This parameter is the time-threshold limit used to regulate the amount of change data that the Capture and Apply programs process during a subscription cycle. You can specify the time-threshold limit using a two-part value. The first part is a number; the second part is the unit of time: ***MIN** Minutes ***HOUR** Hours ***DAY** Days ***WEEK** Weeks You can specify combinations of numbers with units of time. For example, ((1 *WEEK) (2 *DAY) (35 *MIN)) specifies a time interval of one week, two days, and 35 minutes. If you specify multiple occurrences of the same unit of time, the last occurrence is used. The default is zero (0), which indicates that all of the change data is to be applied.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
SQLBEFORE	Specifies the SQL statements that run before the Apply program refreshes the target table. This parameter has three elements: Element 1: SQL code ***NONE** (default) No SQL statement is specified. *SQL-statement* The SQL statement that you want to run. Ensure that the syntax of the SQL statement is correct. DB2 DataPropagator for iSeries does not validate the syntax. In addition, you must use the proper SQL naming conventions. SQL file references must be in the form of LIBRARY.FILE instead of the system naming convention (LIBRARY/FILE). You can specify up to three SQL statements. Element 2: Server to run on ***TGTSVR** (default) The SQL statement runs at the target server on which the target table is located. ***SRCSVR** The SQL statement runs at the Capture control server on which the source table is located. Element 3: Allowed SQLSTATE values ***NONE** (default) Only an SQLSTATE value of 00000 is considered successful. *SQL-states* A list of one to ten allowable SQLSTATE values. Separate the SQLSTATE values with spaces. An SQLSTATE value is a five-digit hexadecimal number ranging from 00000 to FFFFF. The SQL statement is successful if it completes with an SQLSTATE value of 00000 or with one of the allowable SQLSTATE values that you listed.

Table 41. ADDDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
SQLAFTER	Specifies SQL statements that run after the Apply program refreshes the target table. This parameter has three elements: Element 1: SQL code ***NONE** (default) No SQL statement is specified. *SQL-statement* The SQL statement that you want to run. Ensure that the syntax of the SQL statement is correct. DB2 DataPropagator for iSeries does not validate the syntax. In addition, you must use the proper SQL naming conventions. SQL file references must be in the form of LIBRARY.FILE instead of the system naming convention (LIBRARY/FILE). You can specify up to three SQL statements. Element 2: Server to run on ***TGTSVR** (default) The SQL statement runs at the target server on which the target table is located. Element 3: Allowed SQLSTATE values ***NONE** (default) Only an SQLSTATE value of 00000 is considered successful. *SQL-states* A list of one to ten allowable SQLSTATE values. Separate the SQLSTATE values with spaces. An SQLSTATE value is a five-digit hexadecimal number ranging from 00000 to FFFFF. The SQL statement is successful if it completes with an SQLSTATE value of 00000 or with one of the allowable SQLSTATE values that you listed.

Examples for ADDDPRSUB

The following examples illustrate how to use the **ADDDPRSUB** command.

Example 1

To create a subscription set named SETHR under the AQHR Apply qualifier:

```
ADDDPRSUB APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/EMPLOYEE)
  TGTTBL(TGTLIB/TGTEMPL)
```

This subscription set, which contains one subscription-set member, replicates data from the registered source table named EMPLOYEE under the HR library to the target table named TGTEMPL under the TGTLIB library.

Example 2

To create a subscription set named SETHR with only two columns, EMPNO (the key) and NAME, from the registered source table named EMPLOYEE and replicate these columns to an existing target table named TGTEMPL:

```
ADDDPRSUB APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/EMPLOYEE)
  TGTTBL(TGTLIB/TGTEMPL) CRTTGTTBL(*NO) COLUMN(EMPNO NAME) KEYCOL(EMPNO)
```

Example 3

To create a subscription set named SETHR with data from the registered source table named EMPLOYEE and to replicate this data to a replica type target table named TGTREPL:

```
ADDDPRSUB APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/EMPLOYEE)
  TGTTBL(TGTLIB/TGTREPL) TGTTYPE(*REPLICA)
```

Example 4

To create a subscription set named NOMEM with no subscription-set members:

```
ADDDPRSUB APYQUAL(AQHR) SETNAME(NOMEM) SRCTBL(*NONE) TGTTBL(*NONE)
```

Related tasks:

- Chapter 4, "Subscribing to sources for SQL replication," on page 57

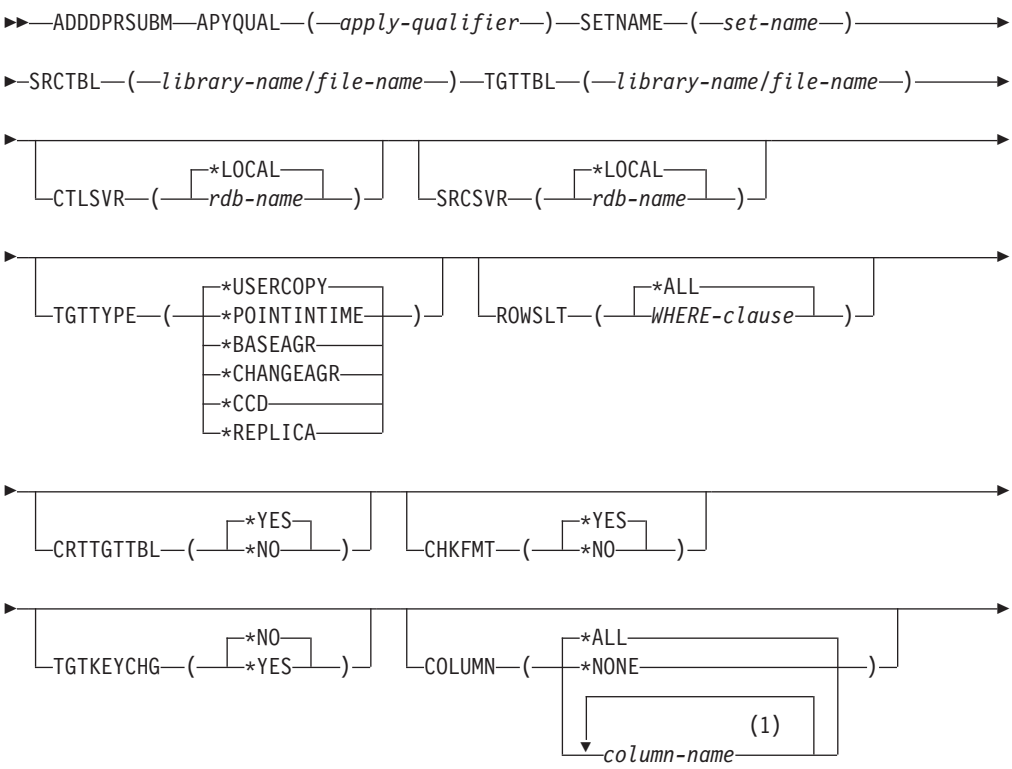
ADDDPRSUBM: Adding a DPR subscription-set member (OS/400)

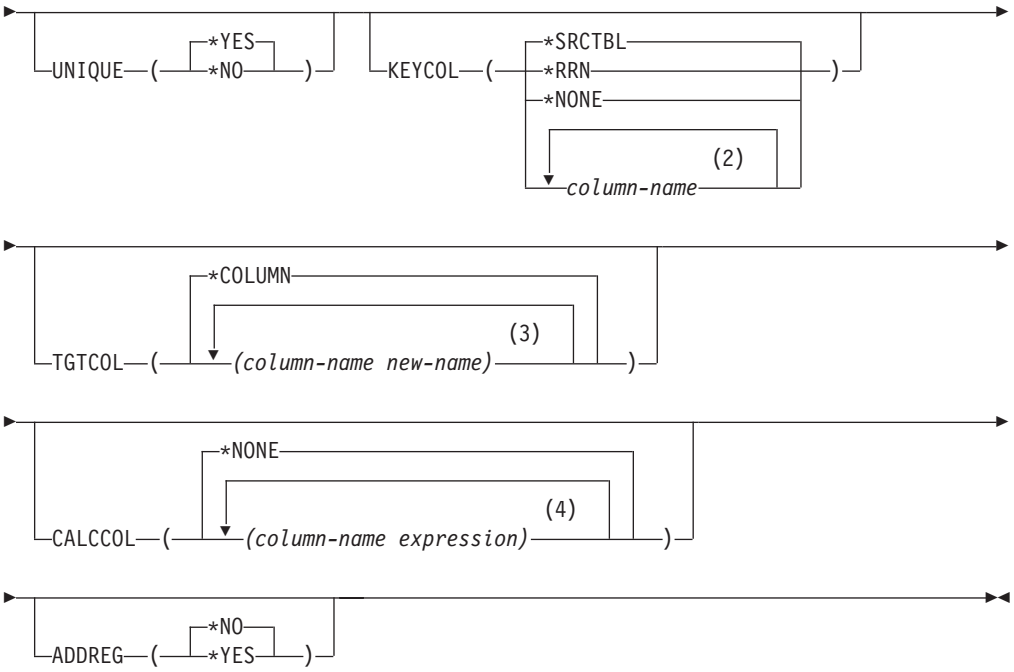
Use the Add DPR subscription-set member (**ADDDPRSUBM**) command to add a member to an existing subscription set. You can create the subscription set with the **ADDDPRSUB** command, with the system commands on UNIX, Windows, or z/OS, or through the Replication Center. All the source tables in the subscription set must already be journaled and must already be registered before you can use this command.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To add a member to a subscription set using the ADDDPRSUBM command:





Notes:

1 You can specify up to 300 column names.

2 You can specify up to 120 column names.

3 You can specify up to 300 column names.

4 You can specify up to 100 column names and expressions.

Table 42 lists the invocation parameters.

Table 42. ADDDPRSUBM command parameter definitions for OS/400

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier that identifies which Apply program processes this subscription set. Subscription sets under an Apply qualifier run in a separate job. This parameter is required. *apply-qualifier* The name of the Apply qualifier.
SETNAME	Specifies the name of the subscription set. This parameter is required. *set-name* The name of the subscription set. The subscription-set name that you enter must be unique for the specified Apply qualifier or the **ADDDPRSUBM** command produces an error. Because the Apply program handles the set of target tables as a group, when one target table fails for any reason, the entire set fails.
SRCTBL	Specifies the name of the table that is the source for this subscription-set member. You must register this table at the Capture control server before this table can become a member of a subscription set. This parameter is required. *library-name/file-name* The qualified name of the source table.

Table 42. ADDDPRSUBM command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TGTTBL	Specifies the name of the target table for this subscription-set member. The target table is automatically created if you set the **CRTTGTTBL** parameter to *YES and the target table does not already exist. This parameter is required. *library-name/file-name* The qualified name of the target table.
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL** (default) The Apply control tables reside locally (on the machine from which you are running the **ADDDPRSUBM** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.
SRCSVR	Specifies the relational database name of the system that contains the Capture control tables. ***LOCAL** (default) The source table is registered on the local machine (the machine from which you are running the **ADDDPRSUBM** command). *rdb-name* The name of the relational database where the Capture control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.
TGTTYPE	Specifies the target table type. These are DB2 replication terms that describe the contents of the target table. After you create a target table as one of these types, you can use this parameter value on the **SRCTBL** parameter of the Add DPR Registration (**ADDDPRREG**) command to register this target table as a source table. ***USERCOPY** (default) The target table is a user copy, which is a target table with content that matches all or part of the content of a source table. A user copy is handled like a point-in-time table but does not contain any of the DB2 DataPropagator for iSeries system columns that are present in the point-in-time target table. This value is not valid when a value of *RRN is specified on the **KEYCOL** parameter. The table that you specified with the **SRCTBL** parameter must be one of the following types: user database, point-in-time table, or consistent-change data (CCD). **Important:** If the target table already exists, DB2 DataPropagator for iSeries does not automatically journal changes to it. You must start journaling outside of DB2 DataPropagator for iSeries.

Table 42. ADDDPRSUBM command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TGTTYPE (continued)	***POINTINTIME** The target table is a point-in-time table. A point-in-time table is a target table with content that matches all or part of the content of the source table and includes the DB2 DataPropagator for iSeries system column (IBMSNAP_LOGMARKER), which identifies when a particular row was inserted or updated at the Capture control server. ***BASEAGR** The target table is a base aggregate table, which is a target table that contains data that is aggregated (calculated) from a source table. The source table for a base aggregate target must be either a user table or a point-in-time table. This target table must contain the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER system timestamp columns. ***CHANGEAGR** The table is a change aggregate table, which is a target table that contains data that is aggregated (calculated) based on the contents of a change-data (CD) table. This target table is created with the IBMSNAP_HLOGMARKER and IBMSNAP_LLOGMARKER system timestamp columns. ***CCD** The table is a consistent-change data (CCD) table, which is a target table created from a join of data in the change-data (CD) table and the unit-of-work (UOW) table. A CCD table provides transaction-consistent data for the Apply program and must include the following columns: • IBMSNAP_INTENTSEQ • IBMSNAP_OPERATION • IBMSNAP_COMMITSEQ • IBMSNAP_LOGMARKER ***REPLICA** The target table is a replica table, which is used only for update-anywhere replication. The replica target table receives changes from the master source table, and changes to the replica target table are propagated back to the master source table. A replica table is automatically registered as a source table.

Table 42. ADDDPRSUBM command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
ROWSLT	Specifies the predicates to be placed in an SQL WHERE clause. The Apply program uses these predicates to determine which rows in the change-data (CD) table of the source to apply to the target table. Use this parameter if you want only a subset of the source changes to be replicated to the target table. ***ALL** (default) The Apply program applies all changes in the CD table to the target table. *WHERE-clause* The SQL WHERE clause that specifies which rows from the CD table the Apply program applies to the target table. Do not include the WHERE keyword; it is implied on this parameter. This WHERE clause must be valid on the data server you are using to run the clause. **Note:** The WHERE clause on this parameter is unrelated to any WHERE clauses specified on the SQLBEFORE or SQLAFTER parameters.
CRTTGTTBL	Specifies whether the target table (or view) is created. ***YES** (default) Creates the target table (or view) if it does not exist. Otherwise, the existing table or view becomes the target, and the format of this existing table or view is checked if the value of the **CHKFMT** parameter is set to *YES. An additional index, with the values that you specified by the **UNIQUE** and **KEYCOL** parameters, is created for a target table if no such index currently exists. The command fails if an existing target table contains rows that violate the conditions of the additional index. ***NO** Does not create the target table or view. You must create the table or view with the correct attributes before starting the Apply program. If the table or view exists and you set **CHKFMT** to *YES, the **ADDDPRSUBM** command ensures that the format of the existing table matches the subscription-set definition that you set. If **CHKFMT** is *NO, you must ensure that the format of the existing table matches the subscription-set definition. **Important:** If the table or view already exists, DB2 DataPropagator for iSeries does not automatically journal changes to the existing object. You must start journaling outside of DB2 DataPropagator for iSeries.
CHKFMT	Specifies whether DB2 DataPropagator for iSeries checks the definition of the subscription-set member against the existing target table to ensure that the columns match. This parameter is ignored if the **CRTTGTTBL** parameter is *YES; this parameter is also ignored if the **CRTTGTTBL** parameter is set to *NO and the target table does not exist. ***YES** (default) DB2 DataPropagator for iSeries verifies that the columns defined for this subscription-set member match the columns in the target table. This command fails if a mismatch is detected. ***NO** DB2 DataPropagator for iSeries ignores differences between the subscription-set member and the existing target table. You must ensure that the target table is compatible with the subscription-set member.

Table 42. ADDDPRSUBM command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TGTKEYCHG	Specifies how the Apply program handles updates when changes occur in source columns that are part of the target key columns for the target table. This parameter works in conjunction with the **USEDELINS** parameter on the **ADDDPRREG** command: • If **USEDELINS** is YES and **TGTKEYCHG** is YES, updates are not allowed. • If **USEDELINS** is YES and **TGTKEYCHG** is NO, updates become delete and insert pairs. • If **USEDELINS** is NO and **TGTKEYCHG** is YES, the Apply program handles this condition with special logic. • If **USEDELINS** is NO and **TGTKEYCHG** is NO, the Apply program processes the changes as normal updates. ***NO** (default) Updates to the source table are staged by the Capture program and processed by the Apply program to the target table. ***YES** The Apply program updates the target table based on the before images of the target key column, meaning that the Apply program changes the predicate to the old values instead of the new.
COLUMN	Specifies the columns to be included in the target table. The column names must be unqualified. Choose the column names from the list of column names that you specified on the **CAPCOL** parameter when you registered the source table. If you set the **IMAGE** parameter to *BOTH when registering this table, you can specify before-image column names. The before-image column names are the original column names with a prefix. This prefix is the character that you specified in the **PREFIX** parameter of the **ADDDPRREG** command. ***ALL** (default) All of the columns that you registered in the source are included in the target table. ***NONE** No columns from the source table are included in the target table. You can use *NONE when you want only computed columns in the target table. This value is required if the **CALCCOL** parameter contains summary functions but no grouping is performed. *column-name* The names of up to 300 source columns that you want to include in the target table. Separate the column names with spaces.

Table 42. ADDDPRSUBM command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
UNIQUE	Specifies whether the target table has unique keys as indicated by the **KEYCOL** parameter. ***YES** (default) The target table supports one net change per key; only one row exists in the target table for that key regardless of how many changes are made to the key. This value specifies that the table contains current data rather than a history of changes to the data. A condensed table includes no more than one row for each primary key value in the table and can be used to supply current information for a refresh. ***NO** The target table supports multiple changes per key. The changes are appended to the target table. This value specifies that the table contains a history of changes to the data rather than current data. A non-condensed table includes more than one row for each key value in the table and can be used to supply a history of changes to the data. A non-condensed table cannot supply current data for a refresh.

Table 42. ADDDPRSUBM command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
KEYCOL	Specifies columns that describe the key of the target table. The column names must be unqualified. For *POINTINTIME, *REPLICA, and *USERCOPY target tables (as specified on the **TGTTYPE** parameter), you must identify one or more columns as the target key for the target table. This target key is used by the Apply program to identify each unique row that changes during change-capture replication. ***SRCTBL** (default) The key columns in the target table are the same as those in the source table. The **ADDDPRREG** command uses the key that is specified in the source table if the source table has a key. The following key columns are used: • Key columns that you defined through DDS when creating the table with the Create Physical File (**CRTPF**) command • Primary and unique keys that you defined with the CREATE TABLE and ALTER TABLE SQL statements • Unique keys that you defined with the CREATE INDEX SQL statements If you use a column as a key more than once and with different ordering, the target table key is defined with ascending order. ***RRN** The key column in the target table is the IBMQSQ_RRN column. The target table is created with an IBMQSQ_RRN column, and this column is used as the key. When the Apply program runs, if the source table is a user table and the target table is a point-in-time table or user copy, the IBMQSQ_RRN column in the target table is updated with the relative record number of the associated record in the source table. Otherwise, the IBMQSQ_RRN column in the target table is updated with the value of the IBMQSQ_RRN column in the source table. ***NONE** The target copy does not contain a target key. You cannot specify *NONE if the target table type is *POINTINTIME, *REPLICA, or *USERCOPY. *column-name* The names of the target columns that you want to use as the target key columns. You can specify up to 120 column names. Separate the column names with spaces.

Table 42. ADDDPRSUBM command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TGTCOL	Specifies the new names for all the columns that the Apply program updates in the target table. These names override the column names taken from the source table. The column names must be unqualified. If you specified a value of *NONE for the **COLUMN** parameter, do not use the **TGTCOL** parameter. Use this parameter to give more meaningful names to the target table columns. Specify each source column name and the name of the corresponding column on the target table. ***COLUMN** (default) The target columns are the same as the columns you specified in the **COLUMN** parameter. *column-name* The column names from the source table that you want to change at the target. You can list up to 300 column names. *new-name* The new names of the target columns. You can list up to 300 new column names. If you do not use this parameter, the name of the column on the target table will be the same as the source column name.
CALCCOL	Specifies the list of user-defined or calculated columns in the target table. The column names must be unqualified. Enclose each column name and expression pair in parenthesis. You must specify a column name for each SQL expression. If you want to define any column as an SQL expression without a GROUP BY clause, you must set the **COLUMN** parameter to *NONE. ***NONE** (default) No user-defined or calculated columns are included in the target table. *column-name* The column names of the user-defined or calculated columns in the target table. You can list up to 100 column names. *expression* The expressions for the user-defined or calculated columns in the target table. You can list up to 100 SQL column expressions.
ADDREG	Specifies whether the target table is automatically registered as a source table. Use this parameter to register CCD target type tables. ***NO** (default) The target table is not registered as a source table. DB2 DataPropagator for iSeries ignores this parameter value if the target type is *REPLICA. Replica target tables are always automatically registered as source tables. ***YES** The target table is registered as a source table. This command fails if you already registered the target table. Do not set this parameter to *YES if the target table type is *USERCOPY, *POINTINTIME, *BASEAGR, or *CHANGEAGR. If you set the **CRTTGTTBL** parameter to *NO, you must create the target table before attempting to register it as a source.

Examples for ADDDPRSUBM

The following examples illustrate how to use the **ADDDPRSUBM** command.

Example 1

To add a subscription-set member to a subscription set named SETHR under the AQHR Apply qualifier:

```
ADDDPRSUBM APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/YTDTAX) TGTTBL(TGTHR/TGTTAX)
```

Example 2

To add a subscription-set member with only two columns, AMOUNT and NAME, from the registered source table named YTDTAX and to replicate these columns to an existing target table named TGTTAX:

```
ADDDPRSUBM APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/YTDTAX) TGTTBL(TGTLIB/TGTTAX)
  CRTTGTTBL(*NO) COLUMN(AMOUNT NAME) CHKFMT(*YES)
```

This command verifies that the AMOUNT and NAME columns defined for this subscription-set member match the columns in the target table.

Example 3

To add a subscription-set member to subscription set named SETHR and to replicate this data to a consistent-change data target table named TGTYTD:

```
ADDDPRSUBM APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/YTDTAX) TGTTBL(TGTLIB/TGTYTD)
  TGTTYPE(*CCD) ADDREG (*YES)
```

This command registers the target table as a source table for DB2 DataPropagator for iSeries.

Related tasks:

- Chapter 4, "Subscribing to sources for SQL replication," on page 57

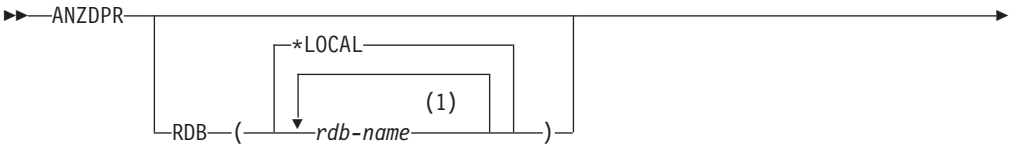
ANZDPR: Operating the Analyzer (OS/400)

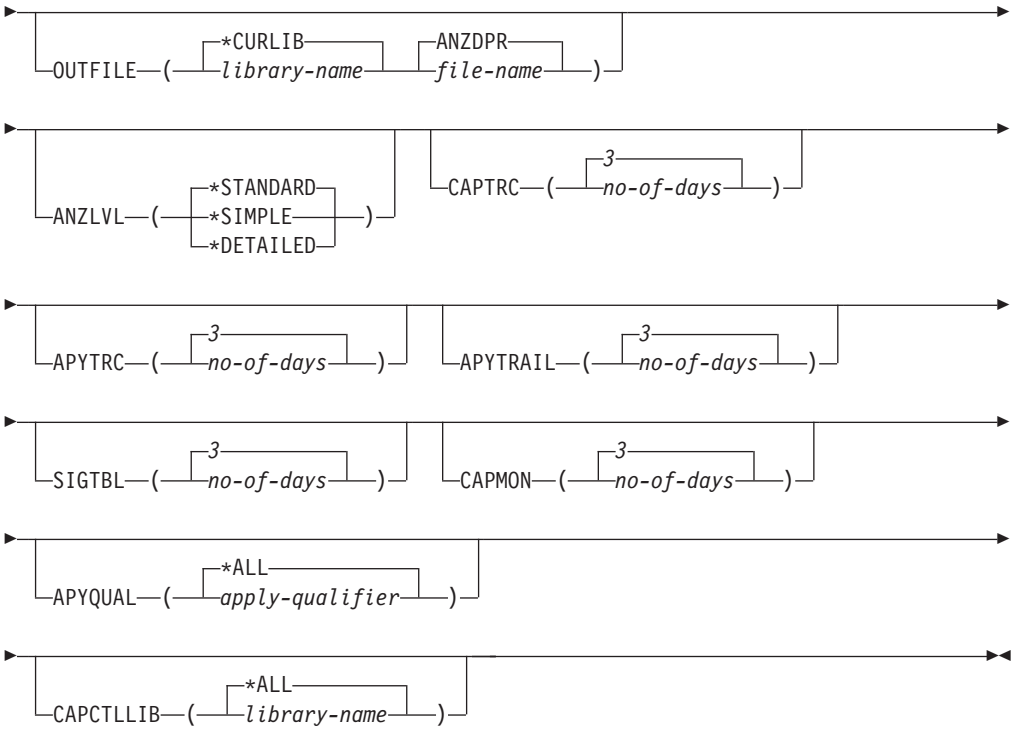
Use the Analyze DPR (**ANZDPR**) command to analyze a failure from a Capture or Apply program, to verify the setup of your replication configuration, or to obtain problem diagnosis and performance tuning information. Run this command after you set up your replication configuration.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To operate the Analyzer using the ANZDPR command:





Notes:

1 You can specify up to 10 databases.

Table 43 lists the invocation parameters.

Table 43. ANZDPR command parameter definitions for OS/400

Parameter	Definition and prompts
RDB	Specifies the databases to be analyzed. ***LOCAL** (default) The database on your local system. *rdb-name* The RDB Directory Entry name, which indicates the database. You can enter up to 10 databases. If you want to analyze multiple databases including the database on your local system, make sure that *LOCAL is the first entry in the list. Also, verify that you can connect to all these databases from your current system.

Table 43. ANZDPR command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
OUTFILE	Specifies the library and file name used to store the analyzer output. This command writes the output to an HTML file. ***CURLIB** (default) The current library. *library-name* The name of the library. **ANZDPR** (default) The output is written to an HTML file named ANZDPR. *file-name* The name of the HTML output file. If the file name already exists, the file is overwritten. If the file name does not exist, the command creates the file with attributes of RCDLEN(512) and SIZE(*NOMAX).
ANZLVL	Specifies the level of analysis to be reported. The level of analysis can be: ***STANDARD** (default) Generates a report that includes the contents of the control tables as well as Capture and Apply program status information. ***SIMPLE** Generates the information in the standard report but excludes subcolumn details. Use this option if you want to generate a smaller report that requires less system resources. ***DETAILED** Generates a report with the most complete analysis. The detailed report includes the information from the standard report in addition to subscription set information.
CAPTRC	Specifies the date range (0 to 30 days) of entries to be reported from the Capture trace (IBMSNAP_CAPTRACE) table. The default is 3. *no-of-days* The number of days to be reported.
APYTRC	Specifies the date range (0 to 30 days) of entries to be reported from the Apply trace (IBMSNAP_APPLYTRACE) table. The default is 3. *no-of-days* The number of days to be reported.
APYTRAIL	Specifies the date range (0 to 30 days) of entries to be reported from the Apply trail (IBMSNAP_APPLYTRAIL) table. The default is 3. *no-of-days* The number of days to be reported.
SIGTBL	Specifies the date range (0 to 30 days) of entries to be reported from the signal (IBMSNAP_SIGNAL) table. The default is 3. *no-of-days* The number of days to be reported.
CAPMON	Specifies the date range (0 to 30 days) of entries to be reported from the Capture monitor (IBMSNAP_CAPMON) table. The default is 3. *no-of-days* The number of days to be reported.

Table 43. ANZDPR command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifiers to be analyzed. ***ALL** (default) All Apply qualifiers are analyzed. *apply-qualifier* The name of the Apply qualifier to be analyzed. You can enter up to 10 Apply qualifiers.
CAPCTLLIB	Specifies the Capture schemas, which are the names of the Capture control libraries that you want to analyze. You can analyze a specific Capture control library, or you can choose the default of *ALL to analyze all the Capture control libraries. ***ALL** (default) All of the Capture control libraries will be analyzed. *library-name* The name of the specific Capture control library that you want to analyze.

Examples for ANZDPR

The following examples illustrate how to use the **ANZDPR** command.

Example 1

To run the Analyzer on both your local database and a remote database named RMTRDB1 using a standard level of analysis:

```
ANZDPR RDB(*LOCAL RMTRDB1) OUTFILE(MYLIB/ANZDPR) ANZLVL(*STANDARD) CAPTRC(1)
  APYTRC(1) APYTRAIL(1) SIGTBL(1) CAPMON(1) APYQUAL(*ALL)
```

This example generates one day of entries from the IBMSNAP_CAPTRACE, IBMSNAP_APPLYTRACE, IBMSNAP_APPLYTRAIL, IBMSNAP_SIGNAL, and IBMSNAP_CAPMON tables for all Apply qualifiers and writes the output to an HTML file named ANZDPR in the library called MYLIB.

Example 2

To run the Analyzer with all default values:

```
ANZDPR
```

Related reference:

- "asnanalyze: Operating the Analyzer" on page 273

CHGDPRCAPA: Changing DPR Capture attributes (OS/400)

Use the Change DPR Capture Attributes (**CHGDPRCAPA**) command to change the global operating parameters that are used by the Capture program and are stored in the Capture parameters (IBMSNAP_CAPPARMS) table. These parameter changes do not take effect until you perform one of the following actions:

- Issue an **INZDPRCAP** command.
- End and then restart the Capture program.

To change the behavior of a running Capture program, see "OVRDPRCAPA: Overriding DPR capture attributes (OS/400)" on page 375.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To change the DPR Capture attributes using the CHGDPRCAPA command:

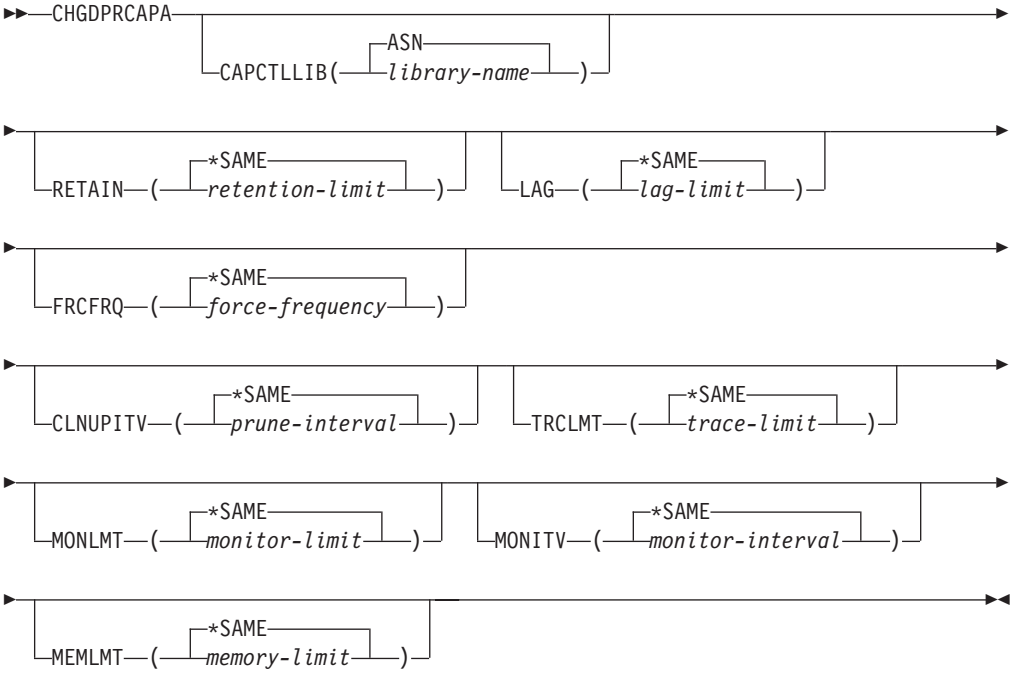


Table 44 lists the invocation parameters.

Table 44. CHGDPRCAPA command parameter definitions for OS/400

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. **ASN** (default) The Capture control tables are in the ASN library. *library-name* The name of a library that contains the Capture control tables.

Table 44. CHGDPRCAPA command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
RETAIN	Specifies the new retention limit, which is the number of minutes that data is retained in the change-data (CD), unit-of-work (UOW), signal (IBMSNAP_SIGNAL), and Apply qualifier cross-reference (IBMSNAP_AUTHTKN) tables before this data is removed. This value is stored in the RETENTION_LIMIT column of the Capture parameters (IBMSNAP_CAPPARMS) table. This value works with the **CLNUPITV** parameter value. When the **CLNUPITV** value is reached, the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN data is removed if this data is older than the retention limit. Ensure that the Apply intervals are set to copy changed information before the data reaches this **RETAIN** parameter value to prevent inconsistent data in your tables. If the data becomes inconsistent, the Apply program performs a full refresh. The default is 10 080 minutes (seven days). The maximum is 35000000 minutes. ***SAME** (default) This value is not changed. *retention-limit* The new retention limit value.
LAG	Specifies the new lag limit, which is the number of minutes that the Capture program can fall behind in processing before restarting. This value is stored in the LAG_LIMIT column of the Capture parameters (IBMSNAP_CAPPARMS) table. When the lag limit is reached (that is, when the timestamp of the journal entry is older than the current timestamp minus the lag limit), the Capture program initiates a cold start for the tables that it is processing for that journal. The Apply program then performs a full refresh to provide the Capture program with a new starting point. The default is 10 080 minutes (seven days). The maximum is 35000000 minutes. ***SAME** (default) This value is not changed. *lag-limit* The new lag limit value.

Table 44. CHGDPRCAPA command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
FRCFRQ	Specifies how often (from 30 to 600 seconds) the Capture program writes changes to the change-data (CD) and unit-of-work (UOW) tables. This value is stored in the COMMIT_INTERVAL column of the Capture parameters (IBMSNAP_CAPPARMS) table. The Capture program makes these changes available to the Apply program either when the buffers are filled or when the **FRCFRQ** time limit expires, whichever is sooner. Use this parameter to make changes more readily available to the Apply program on servers with a low rate of source table changes. The **FRCFRQ** parameter value is a global value used for all defined source tables. Setting the **FRCFRQ** value to a low number can affect system performance. The default is 30 seconds. ***SAME** (default) This value is not changed. *force-frequency* The new commit interval value, which is the number of seconds that the Capture program keeps CD and UOW table changes in buffer space before making these changes available to the Apply program.
CLNUPITV	Specifies the maximum amount of time (in hours) before the Capture program prunes old records from the change-data (CD), unit-of-work (UOW), signal (IBMSNAP_SIGNAL), Capture monitor (IBMSNAP_CAPMON), Capture trace (IBMSNAP_CAPTRACE), and Apply qualifier cross-reference (IBMSNAP_AUTHTKN) tables. This parameter works in conjunction with the **RETAIN** parameter to control pruning of the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN tables, with the **MONLMT** parameter to control pruning of the IBMSNAP_CAPMON table, and with the **TRCLMT** parameter to control pruning of the IBMSNAP_CAPTRACE table. (Use the **STRDPRCAP** command to set the **RETAIN**, **MONLMT**, and **TRCLMT** parameters for a Capture program.) The value of this parameter is automatically converted from hours to seconds and is stored in the PRUNE_INTERVAL column of the Capture parameters (IBMSNAP_CAPPARMS) table. If the PRUNE_INTERVAL column is changed manually (not using the **CHGDPRCAPA** command), you might see changes due to rounding when you prompt using the F4 key. ***SAME** (default) This Capture attribute value is not changed. *prune-interval* The pruning interval expressed as a specific number of hours (1 to 100).

Table 44. CHGDPRCAPA command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TRCLMT	Specifies the trace limit (in minutes). This value is stored in the TRACE_LIMIT column of the Capture parameters (IBMSNAP_CAPPARMS) table. The Capture programs prune any IBMSNAP_CAPTRACE rows that are older than the trace limit. The default is 10 080 minutes (seven days of trace entries). ***SAME** (default) This value is not changed. *trace-limit* The number of minutes of trace data kept in the IBMSNAP_CAPTRACE table after pruning.
MONLMT	Specifies the monitor limit (in minutes). This value is stored in the MONITOR_LIMIT column of the Capture parameters (IBMSNAP_CAPPARMS) table. The Capture program prunes any IBMSNAP_CAPMON rows that are older than the monitor limit. The default is 10 080 minutes (seven days of monitor entries). ***SAME** (default) This value is not changed. *monitor-limit* The number of minutes of monitor data kept in the IBMSNAP_CAPMON table after pruning.
MONITV	Specifies how frequently (in seconds) the Capture program inserts rows into the Capture monitor (IBMSNAP_CAPMON) table. This value is stored in the MONITOR_INTERVAL column of the Capture parameters (IBMSNAP_CAPPARMS) table. The default is 300 seconds (five minutes). ***SAME** (default) This value is not changed. *monitor-interval* The number of seconds between row insertion into the IBMSNAP_CAPMON table. The monitor interval must be at least 120 seconds (two minutes). If you specify a number that is less than 120, this command automatically sets this parameter value to 120.
MEMLMT	Specifies the maximum size (in megabytes) of memory that the Capture journal job can use. This value is stored in the MEMORY_LIMIT column of the Capture parameters (IBMSNAP_CAPPARMS) table. The default is 32 megabytes. ***SAME** (default) This value is not changed. *memory-limit* The maximum number of megabytes for memory.

Examples for CHGDPRCAPA

The following examples illustrate how to use the **CHGDPRCAPA** command.

Example 1

To change the frequency of row insertion to 6 000 seconds (100 minutes) by the Capture program into the IBMSNAP_CAPMON table:

```
CHGDPRCAPA CAPCTLLIB(ASN) MONITV(6000)
```

This frequency value is stored in the IBMSNAP_CAPPARMS table that is located in the default ASN library.

Example 2

To change the retention limit, lag limit, trace limit, and monitor limit in the IBMSNAP_CAPPARMS table located in a Capture control library called LIB1:

```
CHGDPRCAPA CAPCTLLIB(LIB1) RETAIN(6000) LAG(3000) TRCLMT(3000) MONLMT(6000)
```

Example 3

To change the commit interval, which indicates how frequently the Capture program writes changes to the CD and UOW tables:

```
CHGDPRCAPA CAPCTLLIB(ASN) FRCFRQ(360)
```

Related tasks:

- Chapter 9, “Operating the Capture program for SQL replication,” on page 103

CRTDPRTBL: Creating the replication control tables (OS/400)

If your replication control tables are accidentally deleted or corrupted, use the Create DPR Tables **(CRTDPRTBL)** command to create them manually.

Important: The **CRTDPRTBL** command is the only command that you should use to create OS/400 control tables. Do not use the Replication Center to create the control tables.

Restriction: If you create an alternate Capture schema, you must created it in the same Auxiliary Storage Pool (either base or independent) where the ASN library is located.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To create replication control tables using the CRTDPRTBL command:

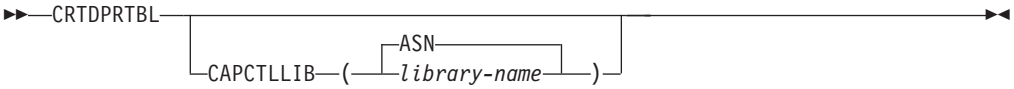


Table 45 on page 361 lists the invocation parameters.

Table 45. CRTDPRTBL command parameter definitions for OS/400

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library where the newly created Capture control tables are placed. **ASN** (default) The Capture control tables are placed in the ASN library. *library-name* The name of the library where the Capture control tables are placed.

Examples for CRTDPRTBL

The following examples illustrate how to use the **CRTDPRTBL** command.

Example 1

To create new replication control tables in the default ASN library:

```
CRTDPRTBL CAPCTLLIB(ASN)
```

Example 2

To create new replication control tables for a Capture schema called DPRSALES:

```
CRTDPRTBL CAPCTLLIB(DPRSALES)
```

Related tasks:

- Chapter 2, "Configuring servers for SQL replication," on page 15

ENDDPRAPY: Stopping Apply (OS/400)

Use the End DPR Apply (**ENDDPRAPY**) command to stop an Apply program on your local system.

You should stop the Apply program before any planned system down time. You might also want to end the Apply program during periods of peak system activity.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To stop an Apply program using the ENDDPRAPY command:

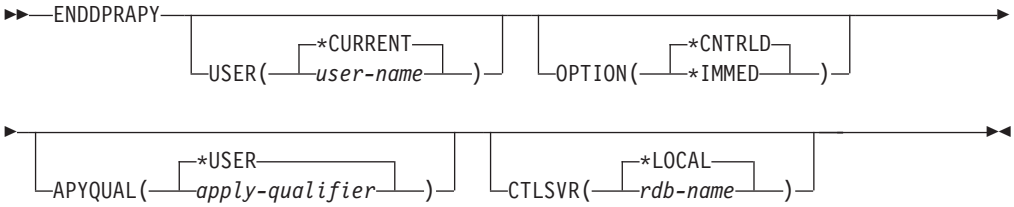


Table 46 on page 362 lists the invocation parameters.

Table 46. ENDDPRAPY command parameter definitions for OS/400

Parameter	Definition and prompts
USER	This parameter is ignored unless the **APYQUAL** parameter has a value of *USER, in which case this parameter specifies the Apply qualifier associated with the Apply program. ***CURRENT** (default) The Apply program of the user associated with the current job. *user-name* The Apply program of the specified user. When prompting on the **ENDDPRAPY** command, you can press the F4 key to see a list of users who defined subscriptions.
OPTION	Specifies how to stop the Apply program. ***CNTRLD** (default) The Apply program completes all of its tasks before stopping. These tasks might take a considerable period of time if the Apply program is completing a subscription set. ***IMMED** The Apply program completes all of its tasks with the **ENDJOB OPTION(*IMMED)** command. The tasks end immediately, without any cleanup. Use this option only after a controlled end is unsuccessful, because it can cause undesirable results. (Unless the Apply program was asleep when you issued the **ENDDPRAPY** command, you should verify the target table contents.) If the Apply program was performing a full refresh to the target table, the target table might be empty as a result of ending the Apply program before the table was refreshed with the source table contents. If the target table is empty, you must force a full refresh for this replication target. You might find that a subscription set is considered IN USE (the STATUS column in the subscription sets (IBMSNAP_SUBS_SET) table has a value of 1). If it is, reset the value to 0 or -1. This allows the subscription set to be run again by the Apply program.
APYQUAL	Specifies the Apply qualifier that is used by the Apply program. ***USER** (default) The user name specified on the **USER** parameter is the Apply qualifier. *apply-qualifier* The name used to group the subscription sets that this Apply program runs. You can specify a maximum of 18 characters for the Apply qualifier name. This name follows the same naming conventions as a relational database name. You identify the subscriptions being run by the records in the subscription sets (IBMSNAP_SUBS_SET) table with this value in the APPLY_QUAL column. When prompting on the **ENDDPRAPY** command, you can press the F4 key to see a list of Apply qualifier names with existing subscriptions.

Table 46. ENDDPRAPY command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL** (default) The Apply control tables reside locally (from the machine on which you are running the **ENDDPRAPY** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name. When prompting on the **ENDDPRAPY** command, you can press the F4 key to choose from the list of databases in the RDB directory.

Usage notes

The **ENDDPRAPY** command uses the value of the **APYQUAL** and **CTLSVR** parameters to search the Apply job (IBMSNAP_APPLY_JOB) table for the job name, job number, and job user for the referenced Apply program, and ends that job.

ENDDPRAPY issues an error message if the following conditions occur:

- If the IBMSNAP_APPLY_JOB table does not exist or is corrupted.
- If there is no record in the IBMSNAP_APPLY_JOB table for the Apply qualifier and control server name.
- If the Apply job already ended.
- If the user ID running the command is not authorized to end the Apply job.

Examples for ENDDPRAPY

The following examples illustrate how to use the **ENDDPRAPY** command.

Example 1

To end the Apply program that uses the AQHR Apply qualifier:

```
ENDDPRAPY OPTION(*CNTRLD) APYQUAL(AQHR)
```

The Apply program ends after all of its tasks are completed.

Example 2

To end the Apply program immediately:

```
ENDDPRAPY OPTION(*IMMED) APYQUAL(AQHR)
```

The tasks of the Apply program end immediately, without any cleanup.

Example 3

To end an Apply program that uses Apply control tables that reside on a relational database named DB1X:

```
ENDDPRAPY OPTION(*CNTRLD) APYQUAL(AQHR) CTLSVR(DB1X)
```

Related tasks:

- Chapter 10, “Operating the Apply program for SQL replication,” on page 121

ENDDPRCAP: Stopping Capture (OS/400)

Use the End DPR Capture (**ENDDPRCAP**) command to stop the Capture program.

Use this command to stop the Capture program before shutting down the system. You might also want to stop the program during periods of peak system use to increase the performance of other programs that run on the system.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To stop the Capture program using the ENDDPRCAP command:

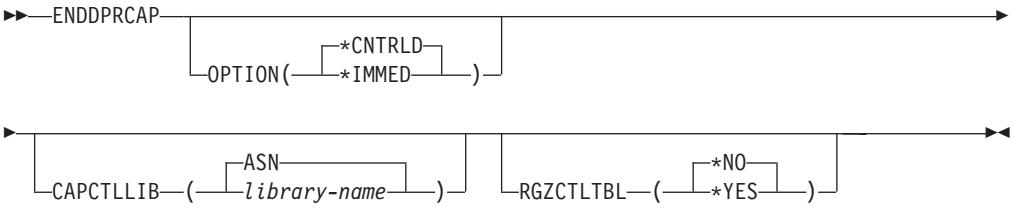


Table 47 lists the invocation parameters.

Table 47. ENDDPRCAP command parameter definitions for OS/400

Parameter	Definition and prompts
OPTION	Specifies how to stop the Capture program. ***CNTRLD** (default) The Capture program stops normally after completing all tasks. The **ENDDPRCAP** command might take longer when you specify the ***CNTRLD** option because the Capture program completes all of its subordinate processes before stopping. ***IMMED** The Capture program stops normally after completing all tasks with the **ENDJOB OPTION(*IMMED)** command.
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables are located. This library includes the register (IBMSNAP_REGISTER) table, which stores the registration information of the source tables. **ASN** (default) The Capture control tables are in the ASN library. The ASN library is the default library. *library-name* The name of a library that contains the Capture control tables.

Table 47. ENDDPRCAP command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
RGZCTLTBL	Specifies whether a Reorganize Physical File Member (**RGZPFM**) command is performed on the control tables (including the change-data (CD) and unit-of-work (UOW) tables) when the Capture program ends. The system does not recover disk space unless the **RGZPFM** command process is performed on the tables. The **RGZPFM** command will not be performed if the control tables are being accessed by an Apply program or by other application programs. ***NO** (default) The **RGZPFM** command is not performed. ***YES** The **RGZPFM** command is performed.

Usage notes

If you use the **ENDJOB** command, temporary objects might be left in the QDP4 library. These objects have the types *DTAQ and *USRSPC, and are named QDP4*nnnnnn*, where *nnnnnn* is the job number of the job that used them. You can delete these objects when the job that used them (identified by the job number in the object name) is not active.

If the job under the Capture control library will not end after issuing this command, use the **ENDJOB** command with *IMMED option to end this job and all the journal jobs running in the DB2 DataPropagator for iSeries subsystem. Do not end Apply jobs running in the same subsystem if you want to end only the Capture program.

In rare cases when the Capture control job ends abnormally, the journal jobs created by Capture control job (which is named according to the **CAPCTLLIB** parameter) might still be left running. The only way to end these jobs is to use the **ENDJOB** command with either the *IMMED or *CNTRLD option.

Examples for ENDDPRCAP

The following examples illustrate how to use the **ENDDPRCAP** command.

Example 1

To end the Capture program, which uses Capture control tables in the ASN library, after all processing tasks are completed:

```
ENDDPRCAP OPTION(*CNTRLD) CAPCTLLIB(ASN) RGZCTLTBL(*NO)
```

Example 2

To end the Capture program immediately for the Capture schema BSN:

```
ENDDPRCAP OPTION(*IMMED) CAPCTLLIB(BSN) RGZCTLTBL(*NO)
```

Example 3

To end the Capture program after all processing tasks are completed and to reorganize the Capture control tables:

```
ENDDPRCAP OPTION(*CNTRLD) CAPCTLLIB(ASN) RGZCTLTBL(*YES)
```

Related tasks:

• Chapter 9, "Operating the Capture program for SQL replication," on page 103

GRTDPRAUT: Authorizing users (OS/400)

The Grant DPR Authority (**GRTDPRAUT**) command authorizes a list of users to the replication control tables, so that the users can run the Capture and Apply programs. For example, the authority requirements for the user who is running the Capture and Apply programs might differ from the authority requirements for the user who defines replication sources and targets.

You must have *ALLOBJ authority to grant authorities.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To authorize users to the replication control tables using the GRTDPRAUT command:

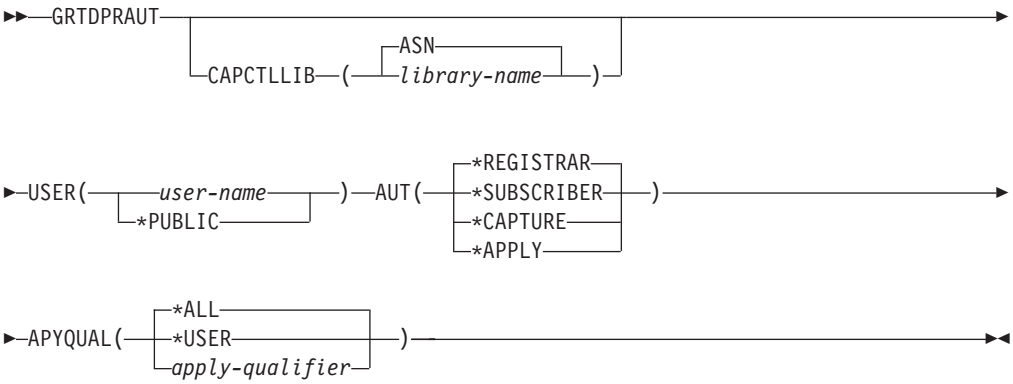


Table 48 lists the invocation parameters.

Table 48. GRTDPRAUT command parameter definitions for OS/400

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the library that contains the replication control tables to which the user is granted authority. **ASN** (default) The Capture control tables reside in the ASN library. *library-name* The name of the library that contains the replication control tables.
USER	Specifies the users who have authority. *user-name* The names of up to 50 users who have authority. ***PUBLIC** Indicates that *PUBLIC authority is granted to the file, but (if insufficient for the task) is used only for those users who have no specific authority, who are not on the authorization list associated with the file, and whose group profile does not have any authority.

Table 48. GRTDPRAUT command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
AUT	Specifies the type of authority being granted. ***REGISTRAR** (default) The users are granted the authorities to define, change, and remove registrations. For a complete list of authorities with AUT(*REGISTRAR), see Table 49 on page 368. ***SUBSCRIBER** The users are granted authority to define, change, and remove subscription sets. For a complete list of authorities with AUT(*SUBSCRIBER), see Table 50 on page 369. ***CAPTURE** The users are granted authority to run the Capture program. For a complete list of authorities granted with AUT(*CAPTURE), see Table 51 on page 370. ***APPLY** The users are granted authority to run the Apply program. The command does not grant authority to any of the objects that reside on other databases accessed by the Apply program. When an Apply program is invoked, the user associated with the DRDA application server job must also be granted *APPLY authority. If the source is an iSeries server, you should run the **GRTDPRAUT** command on the source server system, with the application server job user specified on the **USER** parameter and the Apply qualifier specified on the **APYQUAL** parameter. Authorities are not granted to the target tables unless the target server is the same as the control server and both reside on the system where the command is run. For a complete list of authorities granted with AUT(*APPLY), see Table 52 on page 372.

Table 48. GRTDPRAUT command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier to be used by the user as specified with the **USER** parameter. This parameter is used only when AUT(*APPLY) or AUT(*SUBSCRIBER) is specified. ***ALL** (default) The user is granted authority to run the Apply program or to define and remove subscription sets for *all* Apply qualifiers. ***USER** The users specified on the **USER** parameter are granted authority to the subscription sets with an Apply qualifier that is the same as the user name. *apply-qualifier* The user is granted authority to run the Apply program or define and remove subscription sets for the Apply qualifiers associated with this Apply qualifier. • The user is granted authority to all replication sources, change-data (CD) tables, and consistent-change data (CCD) tables associated with records in the pruning control (IBMSNAP_PRUNCNTL) table that have a value in the APPLY_QUAL column matching the value input with the **APYQUAL** parameter. • The user is granted authority to the subscription sets listed in the subscription members (IBMSNAP_SUBS_MEMBR) table that reside on this system.

Usage notes

You cannot use the **GRTDPRAUT** command while the Capture or Apply programs are running, or when applications that use the source tables are active because authorizations cannot be changed on files that are in use.

The following tables list the authorities granted when you specify:

- AUT(*REGISTRAR)
- AUT*(SUBSCRIBER)
- AUT(*CAPTURE)
- AUT(*APPLY)

on the **GRTDPRAUT** command.

The following table lists the authorities granted when you specify the AUT(*REGISTRAR) parameter on the **GRTDPRAUT** command.

*Table 49. Authorities granted with GRTDPRAUT AUT(*REGISTRAR)*

Library	Object	Type	Authorizations
QSYS	capctllib	*LIB	*USE, *ADD
capctllib[1]	QSQJRN	*JRN	*OBJOPR, *OBJMGT
capctllib[1]	QZS8CTLBLK	*USRSPC	*CHANGE
capctllib[1]	IBMSNAP_REGISTER	*FILE	*OBJOPR, *READ, *ADD, *UPDT, *DLT

*Table 49. Authorities granted with GRTDPRAUT AUT(*REGISTRAR) (continued)*

Library	Object	Type	Authorizations
capctllib[1]	IBMSNAP_REGISTERX	*FILE	*OBJOPR, *READ, *ADD, *UPDT, *DLT
capctllib[1]	IBMSNAP_REGISTERX1	*FILE	*OBJOPR, *READ, *ADD, *UPDT, *DLT
capctllib[1]	IBMSNAP_REGISTERX2	*FILE	*OBJOPR, *READ, *ADD, *UPDT, *DLT
capctllib[1]	IBMSNAP_REG_EXT	*FILE	*OBJOPR, *READ, *ADD, *UPDT, *DLT
capctllib[1]	IBMSNAP_REG_EXTX	*FILE	*OBJOPR, *READ, *ADD, *UPDT, *DLT
capctllib[1]	IBMSNAP_PRUNCNTL	*FILE	*OBJOPR, *READ
capctllib[1]	IBMSNAP_PRUNCNTLX	*FILE	*OBJOPR, *READ
capctllib[1]	IBMSNAP_PRUNCNTLX1	*FILE	*OBJOPR, *READ
capctllib[1]	IBMSNAP_PRUNCNTLX2	*FILE	*OBJOPR, *READ
capctllib[1]	IBMSNAP_PRUNCNTLX3	*FILE	*OBJOPR, *READ
ASN	ASN4B*	*SQLPKG	*USE
ASN	ASN4C*	*SQLPKG	*USE

Notes:

1. The entry *capctllib* in the Library column refers to the value passed to the **CAPCTLLIB** parameter of the **GRTDPRAUT** command; this command updates authority to only one Capture control library at a time.

The following table lists the authorities granted when you specify the AUT(*SUBSCRIBER) parameter on the **GRTDPRAUT** command.

*Table 50. Authorities granted with GRTDPRAUT AUT(*SUBSCRIBER)*

Library	Object	Type	Authorizations
QSYS	ASN	*LIB	*OBJOPR, *READ, *ADD, *EXECUTE
QSYS	capctllib	*LIB	*OBJOPR, *READ, *ADD, *EXECUTE
ASN	IBMSNAP_SUBS_SET	*FILE	*CHANGE
ASN	IBMSNAP_SUBS_COLS	*FILE	*CHANGE
ASN	IBMSNAP_SUBS_EVENT	*FILE	*CHANGE
ASN	IBMSNAP_SUBS_STMTS	*FILE	*CHANGE
ASN	IBMSNAP_SUBS_MEMBR	*FILE	*CHANGE
capctllib[1]	IBMSNAP_REGISTER	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REG_EXT	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE

*Table 50. Authorities granted with GRTDPRAUT AUT(*SUBSCRIBER) (continued)*

Library	Object	Type	Authorizations
capctllib[1]	IBMSNAP_PRUNCNTL	*FILE	*OBJOPR, *READ, *DLT, *ADD, *EXECUTE
capctllib[1]	IBMSNAP_PRUNCNTLX	*FILE	*USE
ASN	ASN4A*	*SQLPKG	*USE
ASN	ASN4U*	*SQLPKG	*USE

Notes:

1. The entry *capctllib* in the Library column refers to the value passed to the **CAPCTLLIB** parameter of the **GRTDPRAUT** command; this command updates authority to only one Capture control library at a time.

The following table lists the authorities granted when you specify the AUT(*CAPTURE) parameter on the **GRTDPRAUT** command.

*Table 51. Authorities granted with GRTDPRAUT AUT(*CAPTURE)*

Library	Object	Type	Authorizations
QSYS	capctllib	*LIB	*OBJOPR, *OBJMGT, *READ, *EXECUTE
QSYS	QDP4	*LIB	*OBJOPR, *ADD, *READ, *EXECUTE
capctllib[1]	QZSN	*MSGQ	*CHANGE
capctllib[1]	IBMSNAP_REGISTER	*FILE	*OBJOPR, *OBJMGT, *READ, *ADD, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REGISTERX	*FILE	*OBJOPR, *OBJMGT, *READ, *ADD, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REGISTERX1	*FILE	*OBJOPR, *OBJMGT, *READ, *ADD, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REGISTERX2	*FILE	*OBJOPR, *OBJMGT, *READ, *ADD, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REG_EXT	*FILE	*OBJOPR, *OBJMGT, *READ, *ADD, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REG_EXTX	*FILE	*OBJOPR, *OBJMGT, *READ, *ADD, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_PRUNCNTL	*FILE	*OBJOPR, *OBJMGT, *READ, *UPD, *EXECUTE

*Table 51. Authorities granted with GRTDPRAUT AUT(*CAPTURE) (continued)*

Library	Object	Type	Authorizations
capctllib[1]	IBMSNAP_PRUNCNTLX	*FILE	*OBJOPR, *OBJMGT, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_PRUNCNTLX1	*FILE	*OBJOPR, *OBJMGT, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_PRUNCNTLX2	*FILE	*OBJOPR, *OBJMGT, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_PRUNCNTLX3	*FILE	*OBJOPR, *OBJMGT, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_CAPTRACE	*FILE	*CHANGE
capctllib[1]	IBMSNAP_CAPTRACEX	*FILE	*CHANGE
capctllib[1]	IBMSNAP_RESTART	*FILE	*CHANGE
capctllib[1]	IBMSNAP_RESTARTX	*FILE	*CHANGE
capctllib[1]	IBMSNAP_AUTHTKN	*FILE	*CHANGE
capctllib[1]	IBMSNAP_AUTHTKNX	*FILE	*CHANGE
capctllib[1]	IBMSNAP_UOW	*FILE	*OBJOPR, *OBJMGT, *READ, *UPD, *DLT, *ADD, *EXECUTE
capctllib[1]	IBMSNAP_UOW_IDX	*FILE	*CHANGE
capctllib[1]	IBMSNAP_PRUNE_SET	*FILE	*CHANGE
capctllib[1]	IBMSNAP_PRUNE_SETX	*FILE	*CHANGE
capctllib[1]	IBMSNAP_CAPPARMS	*FILE	*READ, *EXECUTE
capctllib[1]	IBMSNAP_SIGNAL	*FILE	*CHANGE
capctllib[1]	IBMSNAP_SIGNALX	*FILE	*CHANGE
capctllib[1]	IBMSNAP_CAPMON	*FILE	*CHANGE
capctllib[1]	IBMSNAP_CAPMONX	*FILE	*CHANGE
capctllib[1]	IBMSNAP_PRUNE_LOCK	*FILE	*CHANGE
ASN	ASN4B*	*SQLPKG	*USE
ASN	ASN4C*	*SQLPKG	*USE
ASN	QZS8CTLBLK	*USRSPC	*CHANGE

Notes:

1. The entry *capctllib* in the Library column refers to the value passed to the **CAPCTLLIB** parameter of the **GRTDPRAUT** command; this command updates authority to only one Capture control library at a time.

The following table lists the authorities granted when you specify the AUT(*APPLY) parameter on the **GRTDPRAUT** command.

*Table 52. Authorities granted with GRTDPRAUT AUT(*APPLY)*

Library	Object	Type	Authorizations
QSYS	ASN	*LIB	*OBJOPR, *READ, *EXECUTE
QSYS	capctllib	*LIB	*OBJOPR, *READ, *EXECUTE
QDP4	QZSNAPV2	*PGM	*OBJOPR, *READ, *OBMGT, *OBJALTER, *EXECUTE
capctllib[1]	IBMSNAP_REGISTER	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REGISTERX	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REGISTERX1	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REGISTERX2	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REGISTER_EXT	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_REGISTER_EXTX	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
capctllib[1]	IBMSNAP_SIGNAL	*FILE	*OBJOPR, *READ, *UPD, *ADD, *EXECUTE
capctllib[1]	IBMSNAP_SIGNALX	*FILE	*OBJOPR, *READ, *UPD, *ADD, *EXECUTE
capctllib[1]	IBMSNAP_PRUNE_LOCK	*FILE	*CHANGE
capctllib[1]	IBMSNAP_UOW	*FILE	*OBJOPR, *READ, *UPD, *ADD, *EXECUTE
capctllib[1]	IBMSNAP_PRUNCNTL	*FILE	*OBJOPR, *READ, *UPD, *ADD, *EXECUTE
capctllib[1]	IBMSNAP_AUTHTKN	*FILE	*OBJOPR, *READ, *UPD, *ADD, *EXECUTE
capctllib[1]	IBMSNAP_AUTHTKNX	*FILE	*OBJOPR, *READ, *UPD, *ADD, *EXECUTE
ASN	IBMSNAP_SUBS_SET	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
ASN	IBMSNAP_SUBS_SETX	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
ASN	IBMSNAP_APPLYTRAIL	*FILE	*OBJOPR, *READ, *UPD, *ADD, *EXECUTE
ASN	IBMSNAP_APPLYTRACE	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE

*Table 52. Authorities granted with GRTDPRAUT AUT(*APPLY) (continued)*

Library	Object	Type	Authorizations
ASN	IBMSNAP_APPLYTRACX	*FILE	*OBJOPR, *READ, *UPD, *EXECUTE
ASN	IBMSNAP_SUBS_COLS	*FILE	*USE
ASN	IBMSNAP_SUBS_EVENT	*FILE	*USE
ASN	IBMSNAP_SUBS_STMTS	*FILE	*USE
ASN	IBMSNAP_SUBS_MEMBR	*FILE	*USE
ASN	ASN4A*	*SQLPKG	*USE
ASN	ASN4U*	*SQLPKG	*USE
ASN	IBMSNAP_APPLY_JOB	*FILE	*OBJOPR, *READ, *UPD, *ADD, *EXECUTE

Notes:

1. The entry *capctllib* in the Library column refers to the value passed to the **CAPCTLLIB** parameter of the **GRTDPRAUT** command; this command updates authority to only one Capture control library at a time.

Examples for GRTDPRAUT

The following examples illustrate how to use the **GRTDPRAUT** command.

Example 1

To authorize a user named USER1 to define and modify registrations:

```
GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*REGISTRAR)
```

Example 2

To authorize a user named USER1 to define and modify subscription sets:

```
GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*SUBSCRIBER)
```

Example 3

To authorize a user named USER1 to run Capture programs:

```
GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*CAPTURE)
```

Example 4

To authorize a user named USER1 to define and modify existing subscription sets that are associated with Apply qualifier A1:

```
GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*SUBSCRIBER) APYQUAL(A1)
```

Example 5

To authorize a user to run the Apply program on the control server system for all subscription sets associated with Apply qualifier A1, where the target server is the same as the control server:

1. Run the following command on the system where the Apply program will run:
   ```
   GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*APPLY) APYQUAL(A1)
   ```
2. Run the appropriate **GRTDPRAUT** command on the source server system:
   - If the application server job on the source server used by the Apply program runs under user profile USER1, run the following command on the source server systems:
     ```
     GRTDPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*APPLY) APYQUAL(A1)
     ```

- If the application server job on the source server used by the Apply program runs under a different user profile, for example, QUSER, the command is:
  ```
  GRTDPRAUT CAPCTLLIB(ASN) USER(QUSER) AUT(*APPLY) APYQUAL(A1)
  ```

Related tasks:

- Chapter 2, "Configuring servers for SQL replication," on page 15

Related reference:

- "asnpwd: Creating and maintaining password files" on page 299
- "RVKDPRAUT: Revoking authority (OS/400)" on page 384

INZDPRCAP: Reinitializing DPR Capture (OS/400)

Use the Initialize DPR Capture (**INZDPRCAP**) command to initialize the Capture program by directing the Capture program to work with an updated list of source tables.

Source tables under the control of a Capture program can change while the Capture program is running. Use the **INZDPRCAP** command to ensure that the Capture program processes the most up-to-date replication sources.

The Capture program must be running before you can run this command.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To initialize the Capture program using the INZDPRCAP command:

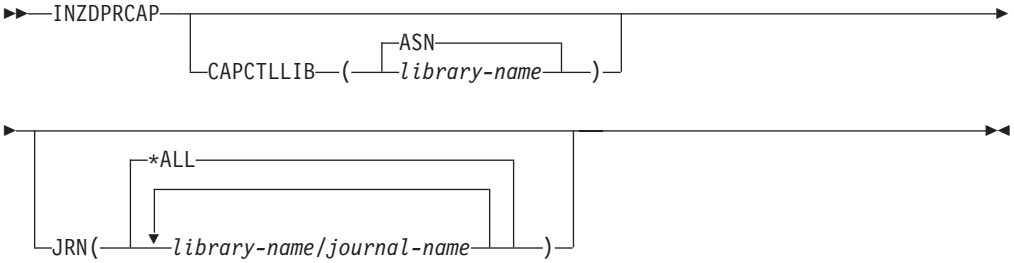


Table 53 on page 375 lists the invocation parameters.

Table 53. INZDPRCAP command parameter definitions for OS/400

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. **ASN** (default) The Capture control tables reside in the ASN library. The ASN library is the default library. *library-name* The name of a library that contains the Capture control tables.
JRN	Specifies a subset of up to 50 journals that you want the Capture program to work with. The Capture program starts processing all the source tables that are currently journaled to this journal. ***ALL** (default) The Capture program works with all the journals. *library-name/journal-name* The qualified name of the journal that you want the Capture program to work with.

Examples for INZDPRCAP

The following examples illustrate how to use the **INZDPRCAP** command.

Example 1

To initialize a Capture program using the QSQJRN journal under a library named TRAINING:

```
INZDPRCAP CAPCTLLIB(ASN) JRN(TRAINING/QSQJRN)
```

The Capture control tables reside in the default ASN schema.

Example 2

To initialize a Capture program that works with all the journals:

```
INZDPRCAP CAPCTLLIB(BSN) JRN(*ALL)
```

The Capture control tables reside in a schema called BSN.

Related tasks:

- Chapter 9, "Operating the Capture program for SQL replication," on page 103

OVRDPRCAPA: Overriding DPR capture attributes (OS/400)

Use the Override DPR Capture attributes (**OVRDPRCAPA**) command to alter the behavior of a running Capture program. This command alters the program behavior by overriding the values that were passed to the Capture program from the Capture parameters (IBMSNAP_CAPPARMS) table or from the **STRDPRCAP** command when the Capture program started.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To override the attributes of a Capture program using the OVRDPRCAPA command:

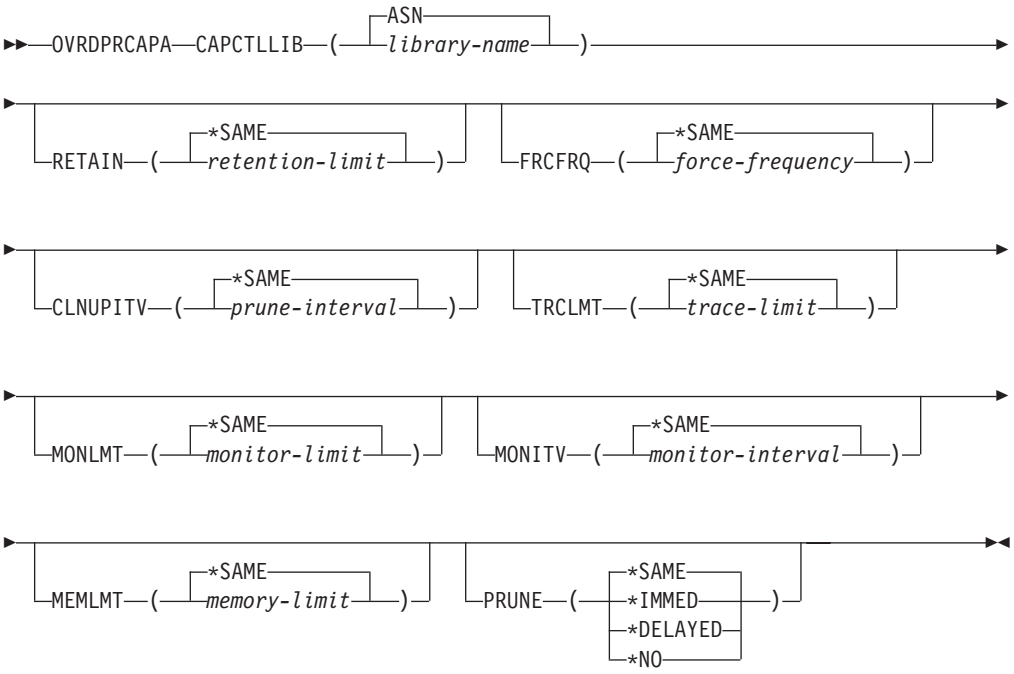


Table 54 lists the invocation parameters.

Table 54. OVRDPRCAPA command parameter definitions for OS/400

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. This library includes the register (IBMSNAP_REGISTER) table, which stores the registration information of the source tables. This parameter is required. **ASN** (default) The Capture control tables reside in the ASN library. *library-name* The name of a library that contains the Capture control tables. You can create this library using the **CRTDPRTBL** command with the **CAPCTLLIB** parameter.

Table 54. OVRDPRCAPA command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
RETAIN	Specifies the number of minutes that data is retained in the change-data (CD), unit-of-work (UOW), signal (IBMSNAP_SIGNAL), and Apply qualifier cross-reference (IBMSNAP_AUTHTKN) tables before the data is removed. This value works with the **CLNUPITV** parameter value from the Start DPR Capture (**STRDPRCAP**) command. First, the Capture program deletes any CD, UOW, IBMSNAP_SIGNAL, or IBMSNAP_AUTHTKN rows that are older than the oldest currently running Apply program. Then, a new or remaining row from the CD, UOW, IBMSNAP_SIGNAL, or IBMSNAP_AUTHTKN table is subsequently deleted when its age reaches the value of the **RETAIN** parameter. Ensure that the Apply intervals are set to copy changed information before the data reaches this **RETAIN** parameter value to prevent inconsistent data in your tables. If the data becomes inconsistent, the Apply program performs a full refresh. The default is 10 080 minutes (seven days). The maximum is 35000000 minutes. ***SAME** (default) This value is not changed. *retention-limit* The new retention limit value.
FRCFRQ	Specifies how often (from 30 to 600 seconds) the Capture program writes changes to the change-data (CD) and unit-of-work (UOW) tables. The Capture program makes these changes available to the Apply program either when the buffers are filled or when the **FRCFRQ** time limit expires, whichever is sooner. This parameter value affects the amount of time that it takes for the Capture program to respond to changes from the Initialize DPR Capture (**INZDPRCAP**) command. Use this parameter to make changes more readily available to the Apply program on servers with a low rate of source table changes. The **FRCFRQ** parameter value is a global value used for all registered source tables. Setting the **FRCFRQ** value to a low number can affect system performance. The default is 30 seconds. ***SAME** (default) This value is not changed. *force-frequency* The new number of seconds that the Capture program keeps CD and UOW table changes in buffer space before making these changes available to the Apply program.

Table 54. OVRDPRCAPA command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
CLNUPITV	Specifies the maximum amount of time (in hours) before the Capture program prunes old records from the change-data (CD), unit-of-work (UOW), signal (IBMSNAP_SIGNAL), Capture monitor (IBMSNAP_CAPMON), Capture trace (IBMSNAP_CAPTRACE), and Apply qualifier cross-reference (IBMSNAP_AUTHTKN) tables. This parameter works with the **RETAIN** parameter to control pruning of the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN tables, with the **MONLMT** parameter to control pruning of the IBMSNAP_CAPMON table, and with the **TRCLMT** parameter to control pruning of the IBMSNAP_CAPTRACE table. (Use the **STRDPRCAP** command to set the **RETAIN**, **MONLMT**, and **TRCLMT** parameters for a Capture program.) The value of the **CLNUPITV** parameter is automatically converted from hours to seconds and is stored in the PRUNE_INTERVAL column of the Capture parameters (IBMSNAP_CAPPARMS) table. ***SAME** (default) This Capture attribute value is not changed. *prune-interval* The pruning interval expressed as a specific number of hours (1 to 100).
TRCLMT	Specifies the trace limit, which indicates how frequently the Capture trace (IBMSNAP_CAPTRACE) table is pruned. ***SAME** (default) The Capture program continues using the current trace limit value. *trace-limit* The number of minutes between each pruning operation of the IBMSNAP_CAPTRACE table.
MONLMT	Specifies the monitor limit, which indicates how frequently the Capture monitor (IBMSNAP_CAPMON) table is pruned. ***SAME** (default) The Capture program continues using the current monitor limit value. *monitor-limit* The number of minutes between each pruning operation of the IBMSNAP_CAPMON table.
MONITV	Specifies the monitor interval (in seconds), which indicates how frequently the Capture program inserts rows into the Capture monitor (IBMSNAP_CAPMON) table. ***SAME** (default) The Capture program continues using the current monitor interval value. *monitor-interval* The number of seconds between row insertion into the IBMSNAP_CAPMON table. The monitor interval must be at least 120 seconds (two minutes). If you type a number that is less than 120, the command automatically sets this parameter value to 120.

Table 54. OVRDPRCAPA command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
MEMLMT	Specifies the maximum size (in megabytes) of memory that the Capture journal job can use. ***SAME** (default) The Capture program continues using the current memory limit value. *memory-limit* The maximum number of megabytes for memory.
PRUNE	Use this parameter to change the way that the Capture program prunes rows from the change-data (CD), unit-of-work (UOW), signal (IBMSNAP_SIGNAL), Capture monitor (IBMSNAP_CAPMON), Capture trace (IBMSNAP_CAPTRACE), and Apply qualifier cross-reference (IBMSNAP_AUTHTKN) tables. ***SAME** (default) The Capture program continues using the pruning parameters that you specified when you started the **STRDPRCAP** command. ***IMMED** The Capture program starts pruning the tables immediately, regardless of the value of the **CLNUPITV** parameter that you specified when you started the **STRDPRCAP** command. ***DELAYED** The Capture program prunes the old rows at the end of the specified pruning interval. PRUNE(*DELAYED) does not affect the frequency of pruning if you set the second part of the **CLNUPITV** parameter to *IMMED or *DELAYED on the **STRDPRCAP** command. However, PRUNE(*DELAYED) *does* initiate pruning if you set the second part of the **CLNUPITV** parameter to *NO when you started the **STRDPRCAP** command. ***NO** The Capture program does not initiate pruning. This value overrides the **CLNUPITV** parameter setting from the **STRDPRCAP** command.

Examples for OVRDPRCAPA

The following examples illustrate how to use the **OVRDPRCAPA** command.

Example 1

To change the pruning parameters of the CD, UOW, IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables (which reside under the default ASN library) and to change the IBMSNAP_CAPMON monitor interval and memory limit of Capture journal jobs in a running Capture program:

```
OVRDPRCAPA CAPCTLLIB(ASN) CLNUPITV(12) MONITV(600) MEMLMT(64)
```

Example 2

To initiate pruning of the CD, UOW, IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables, which reside in the BSN library:

```
OVRDPRCAPA CAPCTLLIB(BSN) PRUNE(*IMMED)
```

Related tasks:

- Chapter 9, "Operating the Capture program for SQL replication," on page 103

Related reference:

- "asnccmd: Operating Capture" on page 288

RMVDPRREG: Removing a DPR registration (OS/400)

Use the Remove DPR registration (**RMVDPRREG**) command to remove a single source table from the register (IBMSNAP_REGISTER) table so that the source table is no longer used for replication.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To remove a DPR registration using the RMVDPRREG command:

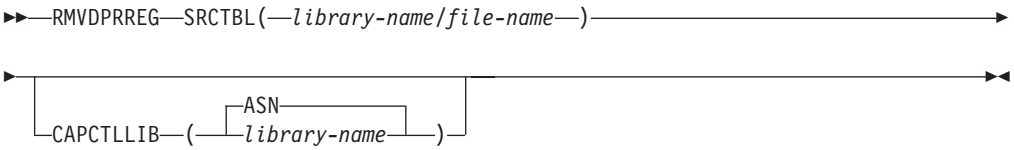


Table 55 lists the invocation parameters.

Table 55. RMVDPRREG command parameter definitions for OS/400

Parameter	Definition and prompts
SRCTBL	Identifies the registration that you want to remove. This is a required parameter. *library-name/file-name* The qualified name of the registered file.
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. **ASN** (default) The Capture control tables are in the ASN library. *library-name* The name of a library containing the Capture control tables.

Examples for RMVDPRREG

The following examples illustrate how to use the **RMVDPRREG** command.

Example 1

To remove the registration for the source table named EMPLOYEE of the HR library in the default ASN Capture schema:

```
RMVDPRREG SRCTBL(HR/EMPLOYEE)
```

Example 2

To remove the registration for the source table named SALES of the DEPT library under a Capture schema called BSN:

```
RMVDPRREG SRCTBL(DEPT/SALES) CAPCTLLIB(BSN)
```

Related tasks:

- Chapter 13, "Making changes to an SQL replication environment," on page 173

RMVDPRSUB: Removing a DPR subscription set (OS/400)

Use the Remove DPR subscription set (**RMVDPRSUB**) command to remove a subscription set. If you set the **RMVMBRS** parameter to *YES, this command removes the subscription set and all of its members.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To remove a subscription set using the RMVDPRSUB command:

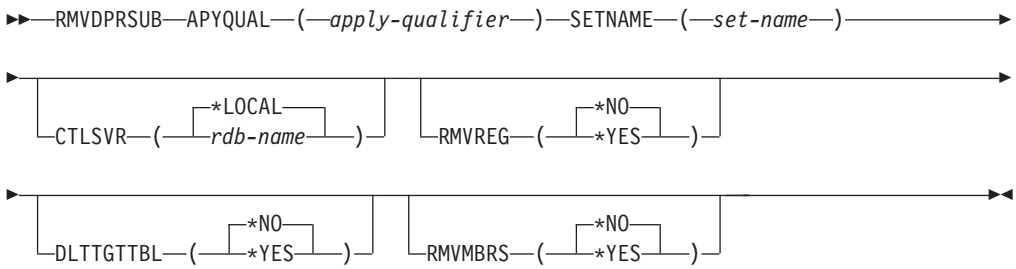


Table 56 lists the invocation parameters.

Table 56. RMVDPRSUB command parameter definitions for OS/400

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier that the Apply program uses to identify the subscription set. This parameter is required. *apply-qualifier* The name of the Apply qualifier.
SETNAME	Specifies the name of the subscription set. This parameter is required. *set-name* The name of the subscription set. You receive an error message if you enter a subscription-set name that does not exist for the specified Apply qualifier.

Table 56. RMVDPRSUB command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL** (default) The Apply control tables reside locally (on the machine from which you are running the **RMVDPRSUB** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.
RMVREG	Specifies whether this command removes the registrations that are associated with the target tables of all the subscription-set members in the subscription set. Use this parameter only if you have set the **RMVMBRS** parameter to *YES. ***NO** (default) The registrations are not removed. ***YES** The registrations are removed.
DLTTGTTBL	Specifies whether this command drops the target tables of the subscription-set members after the subscription set is removed. Use this parameter only if you set the **RMVMBRS** parameter to *YES. ***NO** (default) The target tables are not dropped. ***YES** The target tables are dropped.
RMVMBRS	Specifies whether this command removes the subscription set and all the members in that subscription set. ***NO** (default) The subscription set is not removed if there are existing members in the subscription set. ***YES** The subscription set and all its subscription-set members are removed.

Examples for RMVDPRSUB

The following examples illustrate how to use the **RMVDPRSUB** command.

Example 1

To remove a subscription set named SETHR that contains no subscription-set members:

```
RMVDPRSUB APYQUAL(AQHR) SETNAME(SETHR)
```

Example 2

To remove a subscription set named SETHR and all its subscription-set members:

```
RMVDPRSUB APYQUAL(AQHR) SETNAME(SETHR) RMVMBRS(*YES)
```

Example 3

To remove a subscription set named SETHR, all its subscription-set members, and the associated registrations:

```
RMVDPRSUB APYQUAL(AQHR) SETNAME(SETHR) RMVREG(*YES) RMVMBRS(*YES)
```

Related tasks:

- Chapter 13, "Making changes to an SQL replication environment," on page 173
- Chapter 4, "Subscribing to sources for SQL replication," on page 57

RMVDPRSUBM: Removing a DPR subscription-set member (OS/400)

Use the Remove DPR subscription-set member (**RMVDPRSUBM**) command to remove a single subscription-set member from a subscription set.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To remove a single subscription-set member from a subscription set using the RMVDPRSUBM command:

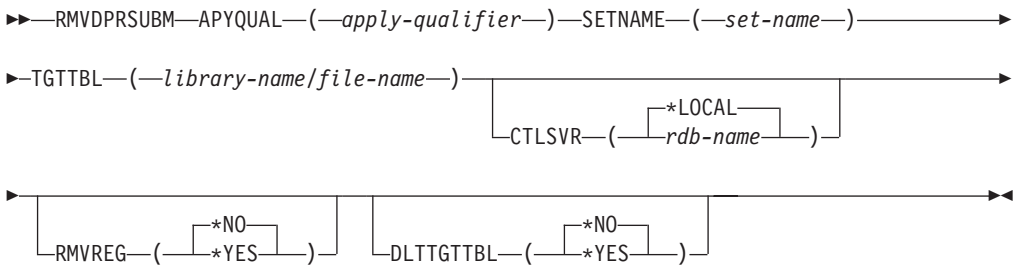


Table 57 lists the invocation parameters.

Table 57. RMVDPRSUBM command parameter definitions for OS/400

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier that the Apply program uses to identify the subscription set. This parameter is required. *apply-qualifier* The name of the Apply qualifier.
SETNAME	Specifies the name of the subscription set. This parameter is required. *set-name* The name of the subscription set. You receive an error message if you enter a subscription-set name that does not exist for the specified Apply qualifier.
TGTTBL	Specifies the target table that is registered for the subscription-set member. This parameter is required. *library-name/file-name* The qualified name of the target table.

Table 57. RMVDPRSUBM command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL** (default) The Apply control tables reside locally (on the machine from which you are running the **RMVDPRSUBM** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name.
RMVREG	Specifies whether this command removes the registration that is associated with the target table for the subscription-set member. ***NO** (default) The registration is not removed. ***YES** The registration is removed.
DLTTGTTBL	Specifies whether this command drops the target table of the subscription-set member after the subscription-set member is removed. ***NO** (default) The target table is not dropped. ***YES** The target table is dropped.

Examples for RMVDPRSUBM

The following examples illustrate how to use the **RMVDPRSUBM** command.

Example 1

To remove a subscription-set member, which uses a target table named EMP, from the SETEMP subscription set on the relational database named RMTRDB1:

```
RMVDPRSUBM APYQUAL(AQHR) SETNAME(SETEMP) TGTTBL(TGTEMP/EMP) CTLSVR(RMTRDB1)
```

Example 2

To remove a subscription-set member from the SETHR subscription set, remove the registration, and then drop the table:

```
RMVDPRSUBM APYQUAL(AQHR) SETNAME(SETHR) TGTTBL(TGTHR/YTDTAX) RMVREG(*YES)
  DLTTGTTBL(*YES)
```

Related tasks:

• Chapter 4, “Subscribing to sources for SQL replication,” on page 57

RVKDPRAUT: Revoking authority (OS/400)

The Revoke DPR Authority (**RVKDPRAUT**) command revokes authority to the replication control tables so that users can no longer define or modify replication sources and subscription sets.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To revoke authority to the replication control tables using the RVKDPRAUT command:

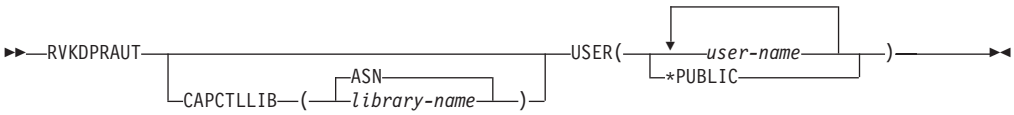


Table 58 lists the invocation parameters.

Table 58. RVKDPRAUT command parameter definitions for OS/400

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library under which user authority is being revoked. **ASN** (default) The Capture control tables reside in the ASN library. *library-name* The name of the library that contains the replication control tables.
USER	Specifies the users whose authority is revoked. This parameter is required. *user-name* Specifies the names of up to 50 users whose authority is revoked. ***PUBLIC** Specifies that authority is revoked from all users without specific authority, who are not on the authorization list, and whose group profile does not have any authority.

Usage notes

The command returns an error message if any of the following conditions exist:

- A specified user does not exist.
- The user running the command is not authorized to the specified user profiles.
- The user running the command does not have permission to revoke authorities to the DB2 DataPropagator for iSeries control tables.
- The DB2 DataPropagator for iSeries control tables do not exist.
- The Capture or Apply programs are running.

Examples for RVKDPRAUT

The following examples illustrate how to use the **RVKDPRAUT** command.

Example 1

To revoke the authority from a user named HJONES to the control tables under the ASN library:

```
RVKDPRAUT CAPCTLLIB(ASN) USER(HJONES)
```

Example 2

To revoke the authority from all users that were not specified in the **GRTDPRAUT** command so that they cannot access the control tables in the ASN library:

```
RVKDPRAUT CAPCTLLIB(ASN) USER(*PUBLIC)
```

Related tasks:

- Chapter 2, "Configuring servers for SQL replication," on page 15

Related reference:

- "GRTDPRAUT: Authorizing users (OS/400)" on page 366

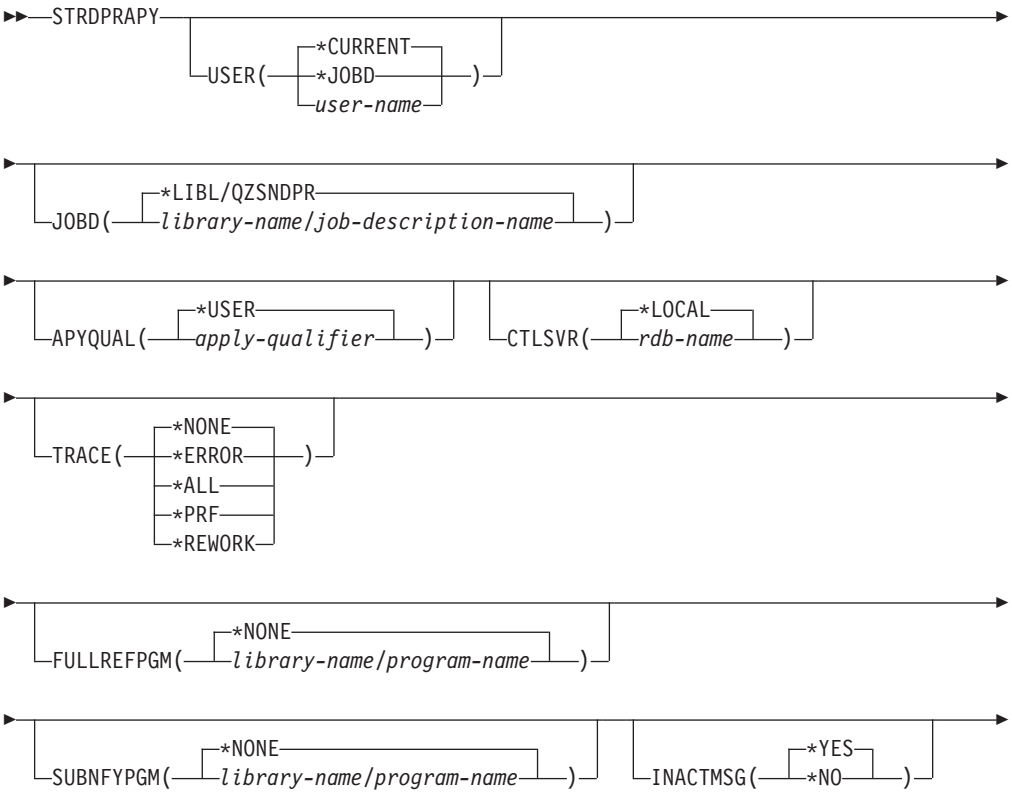
STRDPRAPY: Starting Apply (OS/400)

Use the Start DPR Apply (**STRDPRAPY**) command to start an Apply program on your local system. The Apply program continues to run until you stop it or until it detects an unrecoverable error.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To start the DPR Apply program using the STRDPRAPY command:



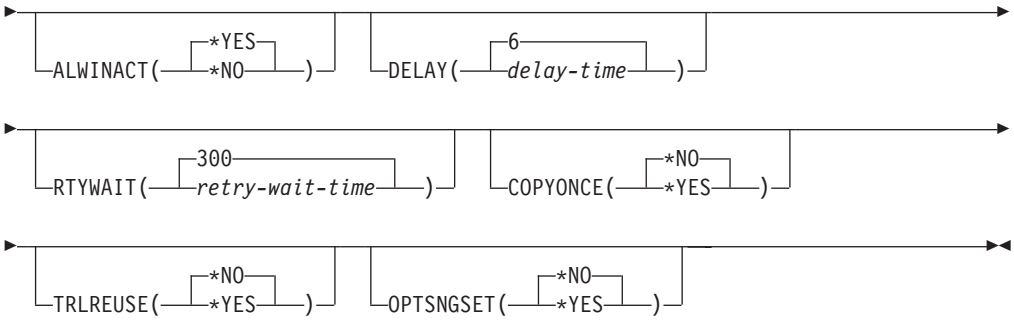


Table 59 lists the invocation parameters.

Table 59. STRDPRAPY command parameter definitions for OS/400

Parameter	Definition and prompts
USER	Specifies the name of the user ID for which to start the Apply program. When you run this command, you must be authorized (have *USE rights) to the specified user profile; the Apply program runs under this specified user profile. The control tables are located on the relational database specified by the **CTLSVR** parameter. The same control tables are used regardless of the value specified on the **USER** parameter. ***CURRENT** (default) The user ID associated with the current job is the same user ID associated with this Apply program. ***JOBD** The user ID specified in the job description associated with this Apply program. The job description cannot specify USER(*RQD). *user-name* The user ID associated with this Apply program. The following IBM-supplied objects are *not* valid on this parameter: QDBSHR, QDFTOWN, QDOC, QLPAUTO, QLPINSTALL, QRJE, QSECOFR, QSPL, QSYS, or QTSTRQS. When prompting on the **STRDPRAPY** command, you can press the F4 key to see a list of users who defined subscription sets.
JOBD	Specifies the name of the job description to use when submitting the Apply program. ***LIBL/QZSNDPR** (default) The default job description provided with DB2 DataPropagator for iSeries. *library-name/job-description-name* The name of the job description used for the Apply program.

Table 59. STRDPRAPY command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier to be used by the Apply program. All subscriptions sets that are grouped together with this Apply qualifier are run by the Apply program. ***USER** (default) The **USER** parameter value that you enter is used as the name of the Apply qualifier. *apply-qualifier* The name used to group the subscription sets that are to be run by this Apply program. You can specify a maximum of 18 characters for the Apply qualifier name. This name follows the same naming conventions as a relational database name. When prompting on the **STRDPRAPY** command, you can press the F4 key to see a list of Apply qualifier names with existing subscription sets.
CTLSVR	Specifies the relational database name of the system that contains the Apply control tables. ***LOCAL** (default) The Apply control tables reside locally (on the machine where you are running the **STRDPRAPY** command). *rdb-name* The name of the relational database where the Apply control tables reside. You can use the Work with RDB Directory Entries (**WRKRDBDIRE**) command to find this name. When prompting on the **STRDPRAPY** command, you can press the F4 key to see a list of available RDB names.
TRACE	Specifies whether the Apply program should generate a trace. The Apply program writes the trace data to a spool file called QPZSNATRC. ***NONE** (default) No trace is generated. ***ERROR** The trace contains error information only. ***ALL** The trace contains error and execution flow information. ***PRF** The trace contains information that can be used to analyze performance at different stages of the Apply program execution. ***REWORK** The trace contains information about rows that were reworked by the Apply program.

Table 59. STRDPRAPY command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
FULLREFPGM	Specifies whether the Apply program is to invoke an exit routine to initialize a target table. When the Apply program is ready to perform a full refresh of a target table, the Apply program invokes the specified exit routine rather than performing the full refresh itself. When a full-refresh exit routine is used by the Apply program, the value of the ASNLOAD column in the Apply trail (IBMSNAP_APPLYTRAIL) table is Y. For examples and more information, see "Refreshing target tables using the ASNLOAD exit routine" on page 135. ***NONE** (default) A full-refresh exit routine is not used. *library-name/program-name* The qualified name of the program that is called by the Apply program performing a full refresh of a target table. For example, to call program ASNLOAD in library DATAPROP, the qualified name is DATAPROP/ASNLOAD.
SUBNFYPGM	Specifies whether the Apply program is to invoke an exit routine when the program finishes processing a subscription set. Input to the exit routine includes the subscription set name, Apply qualifier, completion status, and statistics with the number of rejects. The notify program allows you to examine the unit-of-work (UOW) table to determine when transactions have been rejected and when to take further actions, such as issuing a message or generating an event. For more information, see "Modifying the ASNDONE exit routine (OS/400)" on page 134. ***NONE** (default) An exit routine is not used. *library-name/program-name* The qualified name of the exit routine program called by the Apply program when processing a subscription set. For example, to call program APPLYDONE in library DATAPROP, the qualified name is DATAPROP/APPLYDONE.
INACTMSG	Specifies whether the Apply program is to generate a message whenever the program completes its work and becomes inactive for a period of time. ***YES** (default) The Apply program generates message ASN1044 before beginning a period of inactivity. Message ASN1044 indicates how long the Apply program remains inactive. ***NO** No message is generated.

Table 59. STRDPRAPY command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
ALWINACT	Specifies whether the Apply program can run in an inactive (sleep) state. ***YES** (default) The Apply program sleeps if there is nothing to process. ***NO** If the Apply program has nothing to process, the job that submitted and started the Apply program ends.
DELAY	Specifies the delay time (in seconds) at the end of each Apply program cycle when continuous replication is used. **6** (default) The delay time is six seconds. *delay-time* The delay time, entered as a number between 0 and 6 inclusive.
RTYWAIT	Specifies how long (in seconds) the Apply program is to wait after it encounters an error before it retries the operation that failed. **300** (default) The retry wait time is 300 seconds (five minutes). *retry-wait-time* The wait time, entered as a number between 0 and 35000000 inclusive, before the Apply program retries the failed operation.
COPYONCE	Specifies whether the Apply program executes one copy cycle for each subscription set that is eligible at the time the Apply program is invoked. Then the Apply program terminates. An eligible subscription set meets the following criteria: • (ACTIVATE > 0) in the subscription sets (IBMSNAP_SUBS_SET) table. When the ACTIVATE column value is greater than zero, the subscription set is active indefinitely or is used for a one-time-only subscription processing. • (REFRESH_TYPE = R or B) or (REFRESH_TYPE = E and the specified event occurred). The REFRESH_TYPE column value is stored in the IBMSNAP_SUBS_SET table. The MAX_SYNCH_MINUTES limit from the IBMSNAP_SUBS_SET table and the END_OF_PERIOD timestamp from the subscription events (IBMSNAP_SUBS_EVENT) table are honored if specified. ***NO** (default) The Apply program does not execute one copy cycle for each eligible subscription set. ***YES** The Apply program executes one copy cycle for each eligible subscription set and then terminates.

Table 59. STRDPRAPY command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
TRLREUSE	Specifies whether the Apply program empties the Apply trail (IBMSNAP_APPLYTRAIL) table when the Apply program starts. ***NO** (default) The Apply program does not empty the IBMSNAP_APPLYTRAIL table during program startup. ***YES** The Apply program empties the IBMSNAP_APPLYTRAIL table during program startup.
OPTSNGSET	Specifies whether the performance of the Apply program is optimized if only one subscription set is processed. This parameter does not pertain to replica target tables. If you set this parameter to *YES, the Apply program fetches the members and columns of a subscription set only once and reuses this fetched information when processing the same subscription set in two or more consecutive processing cycles. ***NO** (default) The performance of the Apply program is not optimized if only one subscription set is processed. ***YES** The performance of the Apply program is optimized if only one subscription set is processed. The Apply program reuses the subscription set information during subsequent processing cycles, requiring less CPU resources and improving throughput rates.

Usage notes

You can set up the system to start the subsystem automatically by adding the command that is referred to in the QSTRUPPGM value on your system. If you use the QDP4/QZSNDPR subsystem, it is started as part of the **STRDPRAPY** command processing.

If the relational database (RDB) specified by the **CTLSVR** parameter is a DB2 Universal Database for iSeries database, the tables on the server are found in the ASN library. If the RDB is not a DB2 Universal Database for iSeries database, you can access the tables by using ASN as the qualifier.

Error conditions when starting the Apply program

The **STRDPRAPY** command issues an error message if any of the following conditions occur:

- If the user does not exist.
- If the user running the command is not authorized to the user profile specified on the command or the job description.
- If an instance of the Apply program is already active on the local system for this combination of Apply qualifier and control server.
- If the RDB name specified by the **CTLSVR** parameter is not in the relational database directory.
- If the control tables do not exist on the RDB specified by the **CTLSVR** parameter.

- If no subscription sets are defined for the Apply qualifier specified by the **APYQUAL** parameter.

An Apply program must be started for each unique Apply qualifier in every subscription sets (IBMSNAP_SUBS_SET) table. You can start multiple Apply programs by specifying a different Apply qualifier each time that you issue the **STRDPRAPY** command. These Apply programs will run under the same user profile.

Identifying Apply program jobs

Each Apply program is identified using both the Apply qualifier and the control server names. When run, the job started for the Apply program does not have sufficient external attributes to correctly identify which Apply program is associated with a particular Apply qualifier and control server combination. Therefore, the job is identified in the following way:

- The job is started under the user profile associated with the **USER** parameter.
- The first 10 characters of the Apply qualifier are truncated and become the job name.
- DB2 DataPropagator for iSeries maintains an Apply job (IBMSNAP_APPLY_JOB) table named in the ASN library on the local system. The table maps the Apply qualifier and control server values to the correct Apply program job.
- You can view the job log. The Apply qualifier and control server names are used in the call to the Apply program.

In general, you can identify the correct Apply program job by looking at the list of jobs running in the QZSNDPR subsystem if both:

- The first 10 characters of the Apply qualifier name are unique.
- The Apply program is started for the local control server only.

Examples for STRDPRAPY

The following examples illustrate how to use the **STRDPRAPY** command.

Example 1

To start the Apply program that uses the AQHR Apply qualifier and Apply control tables that reside locally and to generate a trace file that contains error and execution flow information:

```
STRDPRAPY APYQUAL(AQHR) CTLSVR(*LOCAL) TRACE(*ALL)
```

Example 2

To start an Apply program with Apply control tables that reside locally and to specify that the job that started this Apply program automatically end when the Apply program has nothing left to process:

```
STRDPRAPY APYQUAL(AQHR) CTLSVR(*LOCAL) ALWINACT(*NO)
```

Example 3

To start an Apply program that empties the IBMSNAP_APPLYTRAIL table during program startup:

```
STRDPRAPY APYQUAL(AQHR) CTLSVR(*LOCAL) TRLREUSE(*YES)
```

Example 4

To start an Apply program with all default values:

```
STRDPRAPY
```

Related tasks:

- Chapter 10, “Operating the Apply program for SQL replication,” on page 121

Related reference:

- “asnapply: Starting Apply” on page 276

STRDPRCAP: Starting Capture (OS/400)

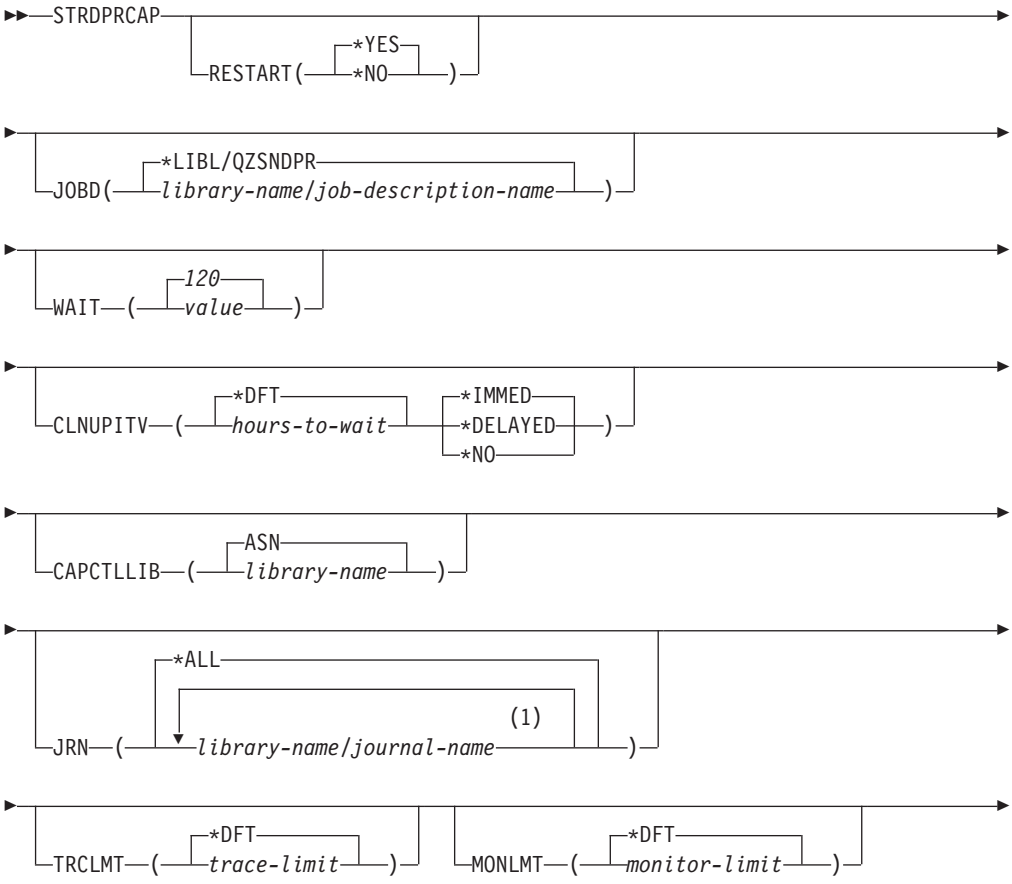
Use the Start DPR Capture (**STRDPRCAP**) command to start capturing changes to OS/400 database tables on iSeries servers. Because this command processes all replication sources in the register (IBMSNAP_REGISTER) table, make sure that you are running this command with the proper authority.

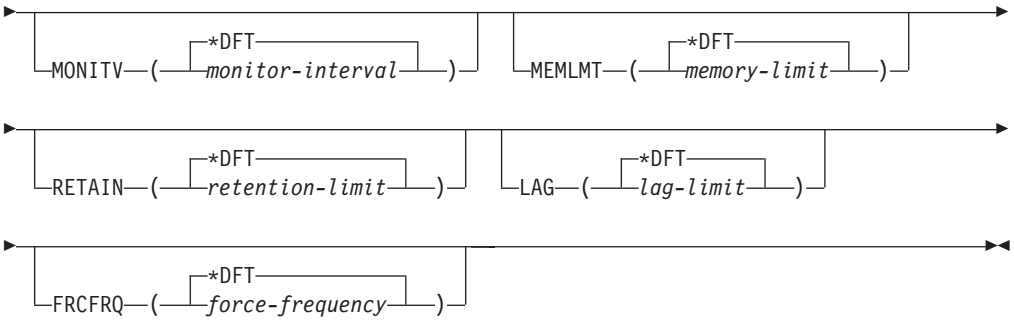
After you start the Capture program, it runs continuously until you stop it or it detects an unrecoverable error.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To start the DPR Capture program using the STRDPRCAP command:





Notes:

1 You can specify up to 50 journals.

Table 60 lists the invocation parameters.

Table 60. STRDPRCAP command parameter definitions for OS/400

Parameter	Definition and prompts
RESTART	Specifies how the Capture program handles warm and cold starts. ***YES** (default) The Capture program continues processing the changes from the point where it was when it ended previously. This is also known as a *warm start* and is the normal mode of operation. ***NO** The Capture program removes all information from the change-data (CD) tables. The Capture program also removes all information from the unit-of-work (UOW) table when you specify JRN(*ALL). All subscriptions for affected source tables are fully refreshed before change capturing resumes. This process is also known as a *cold start*. By specifying RESTART(*NO) and JRN(*library-name/journal-name*), you can cold start the Capture program for specified journals.
JOBD	Specifies the name of the job description to use when submitting the Capture program. ***LIBL/QZSNDPR** (default) Specifies the default job description provided with DB2 DataPropagator for iSeries. *library-name/job-description-name* The name of the job description used for the Capture program.

Table 60. STRDPRCAP command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
WAIT	Specifies the maximum number of seconds (60 to 6 000) to wait before the Capture program checks its status. You can use this value to tune the responsiveness of the Capture program. A low value reduces the time that the Capture program takes before ending or initializing, but can have a negative effect on system performance. A higher value increases the time that the Capture program takes before ending or initializing, but can improve system performance. A value that is too high can result in decreased responsiveness while the Capture program is performing periodic processing. The amount of the decrease in responsiveness depends on the amount of change activity to source tables and the amount of other work occurring on the system. **120** (default) The Capture program waits 120 seconds. *value* The maximum number of seconds that the Capture program waits.
CLNUPITV	Specifies the maximum amount of time (in hours) before the Capture program prunes old records from the change-data (CD), unit-of-work (UOW), signal (IBMSNAP_SIGNAL), Capture monitor (IBMSNAP_CAPMON), Capture trace (IBMSNAP_CAPTRACE), and Apply qualifier cross-reference (IBMSNAP_AUTHTKN) tables. This parameter works with the **RETAIN** parameter to control pruning of the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN tables, with the **MONLMT** parameter to control pruning of the IBMSNAP_CAPMON table, and with the **TRCLMT** parameter to control pruning of the IBMSNAP_CAPTRACE table. (Use the **STRDPRCAP** command to set the **RETAIN**, **MONLMT**, and **TRCLMT** parameters for a Capture program. Use the **CHGDPRCAPA** or **OVRDPRCAPA** command to change these parameter settings.) There are two parts to the **CLNUPITV** parameter: ***DFT** (default) The Capture program uses the value of the PRUNE_INTERVAL column from the Capture parameters (IBMSNAP_CAPPARMS) table. *hours-to-wait* The pruning interval expressed as a specific number of hours (1 to 100). ***IMMED** (default) The Capture program prunes old records at the beginning of the specified interval (or immediately), and at each interval thereafter. ***DELAYED** The Capture program prunes old records at the end of the specified interval, and at each interval thereafter. ***NO** The Capture program does not prune records.

Table 60. STRDPRCAP command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
CAPCTLLIB	Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. **ASN** (default) The default library in which the Capture control tables reside. *library-name* The name of the library in which the Capture control tables reside.
JRN	Specifies a subset of up to 50 journals that you want the Capture program to work with. The Capture program starts processing all the source tables that are currently journaled to this journal. ***ALL** (default) The Capture program starts working with all of the journals that have any source tables journaled to them. *library-name/journal-name* The qualified name of the journal that you want the Capture program to work with. When entering multiple journals, separate the journals with spaces.
TRCLMT	Specifies the trace limit (in minutes). The Capture program prunes any Capture trace (IBMSNAP_CAPTRACE) table rows that are older than the trace limit. The default is 10 080 minutes (seven days of trace entries). ***DFT** (default) The Capture program uses the TRACE_LIMIT column value from the Capture parameters (IBMSNAP_CAPPARMS) table. *trace-limit* The number of minutes of trace data kept in the IBMSNAP_CAPTRACE table after pruning.
MONLMT	Specifies the monitor limit (in minutes). The Capture programs prunes any Capture monitor (IBMSNAP_CAPMON) table rows that are older than the monitor limit. The default is 10 080 minutes (seven days of monitor entries). ***DFT** (default) The Capture program uses the MONITOR_LIMIT column value from the Capture parameters (IBMSNAP_CAPPARMS) table. *monitor-limit* The number of minutes of monitor data kept in the IBMSNAP_CAPMON table after pruning.

Table 60. STRDPRCAP command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
MONITV	Specifies how frequently (in seconds) the Capture program inserts rows into the Capture monitor (IBMSNAP_CAPMON) table. The default is 300 seconds (five minutes). ***DFT** (default) The Capture program uses the MONITOR_INTERVAL column value from the Capture parameters (IBMSNAP_CAPPARMS) table. *monitor-interval* The number of seconds between row insertion into the IBMSNAP_CAPMON table. The monitor interval must be at least 120 seconds (two minutes). If you type a number that is less than 120, this parameter value is set to 120.
MEMLMT	Specifies the maximum size (in megabytes) of memory that the Capture journal job can use. The default is 32 megabytes. ***DFT** (default) The Capture program uses the MEMORY_LIMIT column value from the Capture parameters (IBMSNAP_CAPPARMS) table. *memory-limit* The maximum number of megabytes for memory.
RETAIN	Specifies the new retention limit, which is the number of minutes that data is retained in the change-data (CD), unit-of-work (UOW), signal (IBMSNAP_SIGNAL), and Apply qualifier cross-reference (IBMSNAP_AUTHTKN) tables before it is removed. This value works with the **CLNUPITV** parameter value. When the **CLNUPITV** value is reached, the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN data is removed if this data is older than the retention limit. Ensure that the Apply intervals are set to copy changed information before the data reaches this **RETAIN** parameter value to prevent inconsistent data in your tables. If the data becomes inconsistent, the Apply program performs a full refresh. The default is 10 080 minutes (seven days). The maximum is 35000000 minutes. ***DFT** (default) The Capture program uses the RETENTION_LIMIT column value from the Capture parameters (IBMSNAP_CAPPARMS) table. *retention-limit* The number of minutes that the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN data is retained.

Table 60. STRDPRCAP command parameter definitions for OS/400 (continued)

Parameter	Definition and prompts
LAG	Specifies the new lag limit, which is the number of minutes that the Capture program can fall behind in processing before restarting. When the lag limit is reached (that is, when the timestamp of the journal entry is older than the current timestamp minus the lag limit), the Capture program initiates a cold start for the tables that it is processing in that journal. The Apply program then performs a full refresh to provide the Capture program with a new starting point. The default is 10 080 minutes (seven days). The maximum is 35000000 minutes. ***DFT** (default) The Capture program uses the LAG_LIMIT column value from the Capture parameters (IBMSNAP_CAPPARMS) table. *lag-limit* The number of minutes that the Capture program is allowed to fall behind.
FRCFRQ	Specifies how often (30 to 600 seconds) that the Capture program writes changes to the change-data (CD) and unit-of-work (UOW) tables. The Capture program makes these changes available to the Apply program either when the buffers are filled or when the FRCFRQ time limit expires, whichever is sooner. Use this parameter to make changes more readily available to the Apply program on servers with a low rate of source table changes. The **FRCFRQ** parameter value is a global value used for all defined source tables. Setting the **FRCFRQ** value to a low number can affect system performance. The default is 30 seconds. ***DFT** (default) The Capture program uses the COMMIT_INTERVAL column value from the Capture parameters (IBMSNAP_CAPPARMS) table. *force-frequency* The number of seconds that the Capture program keeps CD and UOW table changes in buffer space before making these changes available to the Apply program.

Usage notes

The **CLNUPITV** parameter on the **STRDPRCAP** command specifies the maximum number of hours that the Capture program waits before pruning old records from the change-data (CD), unit-of-work (UOW), signal (IBMSNAP_SIGNAL), Capture monitor (IBMSNAP_CAPMON), Capture trace (IBMSNAP_CAPTRACE), and Apply qualifier cross-reference (IBMSNAP_AUTHTKN) tables.

You can run the **STRDPRCAP** command manually, or you can run the command automatically as a part of the initial program load (IPL startup program).

If the job description specified with the **JOBD** parameter uses job queue QDP4/QZSNDPR and the DB2 DataPropagator for iSeries subsystem is not active,

the **STRDPRCAP** command starts the subsystem. If the job description is defined to use a different job queue and subsystem, you must start this subsystem manually with the Start Subsystem (**STRSBS**) command either before or after running the **STRDPRCAP** command:

```
STRSBS QDP4/QZSNDPR
```

You can set up the system to start the subsystem automatically by adding the **STRSBS** command to the program that is referred to in the QSTRUPPGM system value on your system.

Restarting Capture using warm or cold starts

The value of the **RESTART** parameter on the **STRDPRCAP** command controls how the Capture program handles warm and cold starts.

Warm start process: Warm start information is saved in most cases. Occasionally, warm start information is not saved. In this case, the Capture program uses the CD tables, UOW table, or the pruning control (IBMSNAP_PRUNCNTL) table to resynchronize to the time that it was stopped.

Automatic cold starts: Sometimes the Capture program automatically switches to a cold start, even if you specified a warm start. On OS/400 systems, cold starts work on a journal-by-journal basis. So, for example, if a journal exceeds the lag limit, all replication sources using that journal are cold-started, whereas replication sources using a different journal are not cold started.

For more information about how the Capture program processes different journal entry types, see Table 114 on page 509.

Examples for STRDPRCAP

The following examples illustrate how to use the **STRDPRCAP** command.

Example 1

To initiate a warm start of a Capture program for two different journals:

```
STRDPRCAP RESTART(*YES) JRN(HR/QSQJRN ACCTS/QSQJRN)
```

Example 2

To start a Capture program for one specified journal:

```
STRDPRCAP CAPCTLLIB(BSN) JRN(MARKETING/QSQJRN)
```

The Capture control tables reside under a library named BSN.

Example 3

To start a Capture program without pruning for two journals:

```
STRDPRCAP RESTART(*YES) CLNUPITV(*DFT *NO) JRN(HR/QSQJRN ACCTS/QSQJRN)
```

Example 4

To start a Capture program for one specified journal under the default Capture control library and to change the default trace limit pruning, monitor limit pruning, IBMSNAP_CAPMON table insertion, and memory limit parameters:

```
STRDPRCAP CAPCTLLIB(ASN) JRN(SALES/QSQJRN) TRCLMT(1440) MONLMT(1440)
  MONITV(3600) MEMLMT(64)
```

Example 5

To initiate a cold start of a Capture program:

```
STRDPRCAP RESTART(*NO)
```

Example 6

To start a Capture program with all default values:

```
STRDPRCAP
```

Related tasks:

- Chapter 9, "Operating the Capture program for SQL replication," on page 103

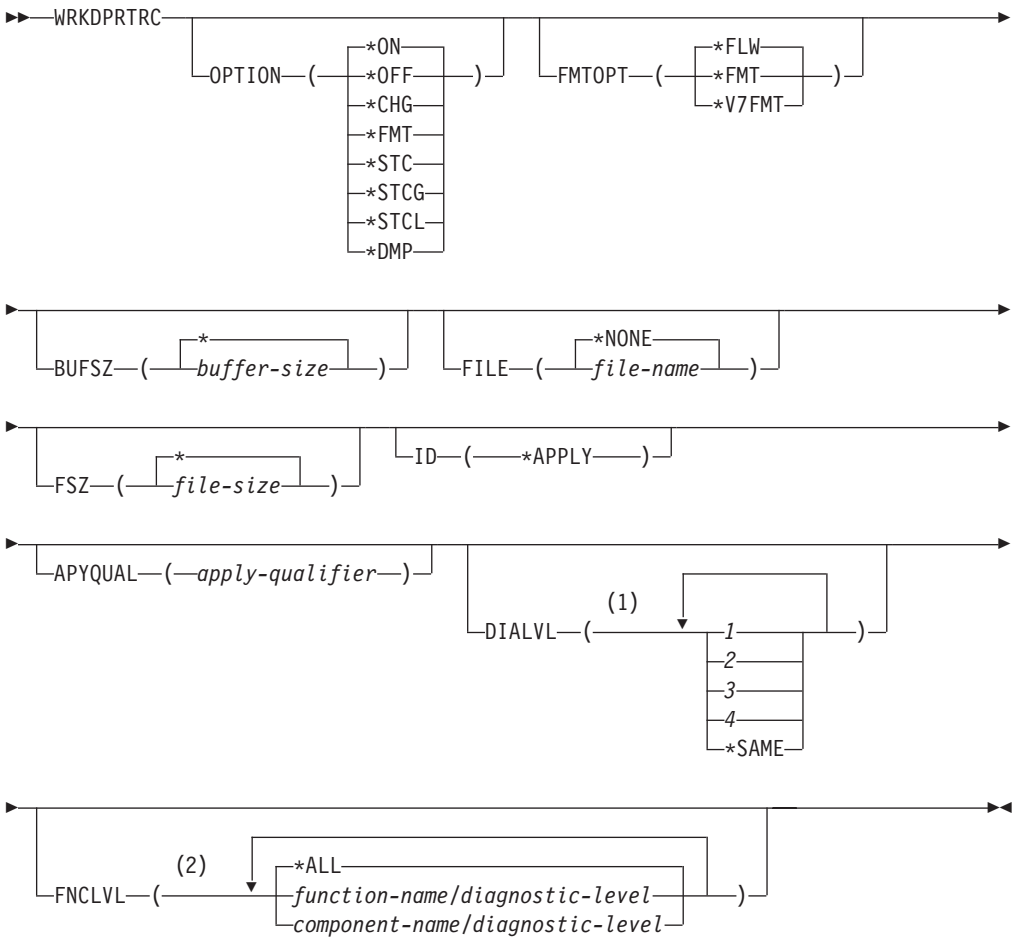
WRKDPRTRC: Using the DPR trace facility (OS/400)

Use the DPR trace (**WRKDPRTRC**) command to run the trace facility. The trace facility logs program flow information for specified Apply programs. You can provide this trace information to IBM Software Support for troubleshooting assistance.

After you type the command name on the command line, you can press the F4 key to display the command syntax.

To display a complete description of this command and all of its parameters, move the cursor to the command at the top of the screen and press the F1 key. To display a description of a specific parameter, place the cursor on that parameter and press the F1 key.

To run the DPR trace facility using the WRKDPRTRC command:



Notes:

1 You can specify multiple values.

2 You can specify up to 20 functions or components.

Table 61 lists the invocation parameters.

Table 61. WRKDPRTRC command parameter definitions for OS/400

Parameter	Definition
OPTION	Specify one trace facility function. ***ON** (default) Turn the trace facility on. This option automatically creates a shared memory segment for tracing. ***OFF** Turn the trace facility off. ***CHG** Change the values of the trace facility parameters. ***FMT** Format the trace facility output from shared memory. ***STC** Display the status of a trace facility. This status information includes the trace version, application version, number of entries, buffer size, amount of buffer used, status code, and program timestamp. This parameter option is equivalent to the stat option of the **asntrc** command used on UNIX, Windows, and z/OS operating systems. ***STCG** Display the status of a trace facility in Replication Center readable format. ***STCL** Display the status of a trace facility with additional version level information. This additional information includes the service levels of each module in the application and appears as long strings of text. This parameter option is equivalent to the statlong option of the **asntrc** command used on UNIX, Windows, and z/OS operating systems. ***DMP** Write the current contents of the trace buffer to a file. When prompting on the **WRKDPRTRC** command, you can press the F4 key to see a list of trace options.

Table 61. WRKDPRTRC command parameter definitions for OS/400 (continued)

Parameter	Definition
FMTOPT	Specifies the options of the format ID and is used with the **OPTION(*FMT)** parameter. ***FLW** (default) Display the flow of the function calls. ***FMT** Display the format of the trace buffer or trace file. Shows all the detailed data. ***V7FMT** Format the trace buffer or trace file information in Version 7 format. When prompting on the **WRKDPRTRC** command, you can press the F4 key to see a list of format options.
BUFSZ	Specifies the size (in bytes) of the trace buffer. You can enter an M, K, or G after the number to indicate megabytes, kilobytes, or gigabytes, respectively. The default is two megabytes. * (default) Use the two megabyte default size. *buffer-size* The buffer size in bytes.
FILE	Specifies whether the trace output is written to a file. ***NONE** (default) The trace output goes to shared memory only. *file-name* The name of the output file. If you are using the **OPTION(*DMP)** parameter, this file name represents the name of a dump file.
FSZ	Specifies the size (in bytes) of the file where the trace data is stored. You can enter an M, K, or G after the number to indicate megabytes, kilobytes, or gigabytes, respectively. The default is two gigabytes. * (default) Use the two gigabyte default size. *file-size* The file size in bytes.
ID	Specifies the type of program to be traced. ***APPLY** (default) An Apply program trace.
APYQUAL	Specifies the name of Apply program to be traced. *apply-qualifier* The name of the Apply qualifier.

Table 61. WRKDPRTRC command parameter definitions for OS/400 (continued)

Parameter	Definition
DIALVL	Specifies the types of trace records to be recorded by the trace facility. Trace records are categorized by a diagnostic mask number: **1** Flow data, which includes the entry and exit points of functions. **2** Basic data, which includes all major events encountered by the trace facility. **3** Detailed data, which includes the major events with descriptions. **4** Performance data. ***SAME** This command uses the diagnostic level settings from the previous trace facility. You can enter one or more diagnostic mask numbers. The numbers that you enter must be in ascending order. Do not type spaces between the numbers. **Important:** The number levels are *not* inclusive. When you start the trace facility, the default is DIALVL(1234). When you subsequently invoke the trace facility, the default is *SAME. When prompting on the **WRKDPRTRC** command, you can press the F4 key to see a list of available diagnostic levels.
FNCLVL	Specifies if a particular function or component identifier is to be traced. ***ALL** (default) All functions and components are included in the trace facility. *function-name/diagnostic-level* The name of the function to be traced and the corresponding diagnostic mask numbers. *component-name/diagnostic-level* The name of the component to be traced and the corresponding diagnostic mask numbers. You can enter up to 20 function or component names.

Examples for WRKDPRTRC

The following examples illustrate how to use the **WRKDPRTRC** command.

Example 1

To start an Apply trace on the Apply qualifier AQ1 for all functions and components with output written to a file called TRCFILE:

```
WRKDPRTRC OPTION(*ON) FILE(TRCFILE) ID(*APPLY) APYQUAL(AQ1)
```

Example 2

To end an Apply trace on the Apply qualifier AQ1:

```
WRKDPRTRC OPTION(*OFF) ID(*APPLY) APYQUAL(AQ1)
```

Example 3

To change an Apply trace on the Apply qualifier AQ1 to diagnostic levels 3 and 4 (detailed and performance data) for all functions and components:

```
WRKDPRTRC OPTION(*CHG) ID(*APPLY) APYQUAL(AQ1) DIALVL(34)
```

Example 4

To display the status of an Apply trace on the Apply qualifier AQ1:

```
WRKDPRTRC OPTION(*STC) ID(*APPLY) APYQUAL(AQ1)
```

Example 5

To display the function calls on the Apply qualifier AQ1 at diagnostic levels 3 and 4:

```
WRKDPRTRC OPTION(*FMT) FMTOPT(*FLW) ID(*APPLY) APYQUAL(AQ1) DIALVL (34)
```

Example 6

To write the Apply trace information of the Apply qualifier AQ1 to a dump file named DMPFILE:

```
WRKDPRTRC OPTION(*DMP) FILE(DMPFILE) ID(*APPLY) APYQUAL(AQ1)
```

Related reference:

• "asntrc: Operating the replication trace facility" on page 309

Chapter 20. Operating the SQL replication programs (z/OS)

This chapter consists of the following sections:

- "Using JCL or system-started tasks to operate the replication programs (z/OS)"
- "Using MVS Automatic Restart Manager (ARM) to automatically restart replication programs (z/OS)" on page 407
- "Migrating your replication environment to data-sharing mode (z/OS)" on page 408

Using JCL or system-started tasks to operate the replication programs (z/OS)

On z/OS, you can operate the Capture program, the Apply program, and the Replication Alert Monitor either with JCL or as system started tasks.

Using JCL to operate replication programs

This section describes how to use JCL to operate the Capture program, Apply program, and the Replication Alert Monitor.

The DB2 DataPropagator V8 samples library contains sample JCL and scripts.

Recommendation: Copy the jobs from the SASNSAMP library to a different library before making changes. See the Program Directory for a complete list of the sample jobs found in the SASNSAMP library.

To start the Capture program on z/OS with JCL:

1. Prepare the JCL for z/OS by specifying the appropriate optional invocation parameters in the PARM field of the Capture job. If you did not set the TZ environment variable in either the system-wide /etc/profile file or in the .profile file in the home directory of the user running the replication program, you must set the TZ and language environment variables in the JCL. For more information about setting the TZ variable, see the *z/OS UNIX System Servives User's Guide*.

   The following example of this line in the invocation JCL includes setting the TZ and LANG variables:

   ```
   //CAPJFA EXEC PGM=ASNCAP, PARM='ENVAR('TZ=PST8PDT','LANG=en_US')/
                             DSN6 cold capture_schema=JFA autostop'
   ```

2. Submit the JCL from TSO or from the MVS console.

To start the Apply program on z/OS with JCL:

Prepare the JCL for z/OS by specifying the appropriate invocation parameters in the PARM field of the Apply job. Customize the JCL to meet your site's requirements.

For z/OS operating systems, an example of this line in the invocation JCL is:

```
//apyasn EXEC PGM=ASNAPPLY,PARM='control_server=CTLDB1
                            DB2_SUBSYSTEM=DSN
                            apply_qual=myqual spillfile=disk'
```

For UNIX and Window operating systems, an example of this line in the invocation JCL is:

```
//apyasn EXEC PGM=ASNAPPLY,PARM='control_server=CTLDB1
                            apply_qual=myqual spillfile=disk'
```

To start the Replication Alert Monitor on z/OS with JCL:

Prepare the JCL for z/OS by specifying the appropriate invocation parameters in the PARM field of the Replication Alert Monitor job. Customize the JCL to meet your site's requirements. A sample of invocation JCL in library SASNSAMP(ASNMON#) is included with the Replication Alert Monitor for z/OS.

An example of this line in the invocation JCL is:

```
//monasn EXEC PGM=ASNMON,PARM='monitor_server=DSN
                          monitor_qual=monqual'
```

where `DSN` is a subsystem name and `monqual` is the monitor qualifier.

To run programs on z/OS with JCL in batch mode:

Customize the JCL in library SASNSAMP for the appropriate program. Table 62 shows which sample job to use to start the specified program:

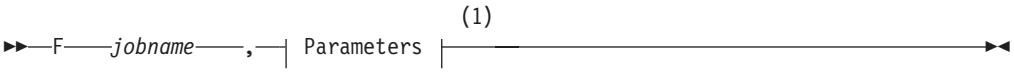
Table 62.

Sample	Program
ASNSTRA	Apply
ASNSTRC	Capture
ASNSTRM	Alert Monitor

Prepare the JCL for z/OS by specifying the appropriate optional invocation parameters in the PARM field of the DPROPR jobs (Capture, Apply, Monitor and Asntrc). Submit the JCL from TSO or from the MVS console.

To modify started programs on z/OS with JCL:

After you start the Capture program, the Apply program, or the Replication Alert Monitor program, you can use the MODIFY command to stop the program or to perform related tasks. You must run the MODIFY command from an MVS console. You can use the abbreviation F, as shown in the following syntax example:



Notes:

1 For descriptions of the parameters, see Chapter 18, "System commands for SQL replication (Linux, UNIX, Windows, z/OS)," on page 271.

Basically, `F` *`jobname`* **`,`** replaces the actual command name: **asnacmd**, **asnccmd**, or **asnmcmd**. For example, to stop the Capture program you would use the following command:

```
F capjfa,stop
```

For information about MODIFY, see *z/OS MVS System Commands*.

Using system-started tasks to operate replication programs

This section describes how to use system-started tasks to operate the Capture program, Apply program, and the Replication Alert Monitor.

Setup to start the Capture for z/OS program as a system-started task:

1. Create a procedure (*procname*) in your PROCLIB.
2. Create an entry in the RACF STARTED class for the *procname*. This entry associates the *procname* with the RACF user ID to be used to start the Capture program. Make sure that the necessary DB2 authorization is granted to this user ID before you start the Capture program.
3. From the MVS system console, run the command **start** *procname*.

The following sample procedure is for the Capture program:

```
//CAPJAYC PROC
//ASNCAP EXEC PGM=ASNCAP,REGION=M,
//PARM='V71A autostop LOGSTDOUT startmode=COLD
//capture_schema=JAY logreuse'
//STEPLIB DD DISP=SHR,DSN=DPROPR.ASN81 .SASNLOAD
//DD DISP=SHR,DSN=SYS1.SCEERUN
//DD DISP=SHR,DSN=DSN7.SDSNLOAD
//CEEDUMP DD SYSOUT=
//SYSPRINT DD SYSOUT=
//SYSTERM DD DUMMY
//
```

Using MVS Automatic Restart Manager (ARM) to automatically restart replication programs (z/OS)

The Capture program, the Apply program, and the Replication Alert Monitor can be used with the MVS Automatic Restart Manager (ARM). ARM is an MVS recovery function that can improve the availability of specific batch jobs or started tasks. When a job or task fails, or the system on which it is running fails, ARM can restart the job or task without operator intervention. ARM uses element names to identify the applications with which it works. Each MVS ARM enabled application generates a unique element name for itself that it uses in all communication with ARM. ARM tracks the element name and has its restart policy defined in terms of element names. For details about setting up ARM, see *z/OS MVS Sysplex Services Guide*, SA22-7617.

Prerequisites:

Ensure that ARM is installed and that the replication programs are set up correctly. If you are going to use ARM with a replication program, ensure that the replication program is APF authorized. For example, to use ARM for the Apply program or the Replication Alert Monitor, you must copy the appropriate load module into an APF authorized library. (The Capture program must be APF authorized regardless of whether or not you are using ARM.)

When you configure ARM, use the following element names for the replication programs:

Capture program
ASNTC*xxxxyyyy*

Apply program
ASNTA*xxxxyyyy*

Replication Alert Monitor
ASNAM*xxxxyyyy*

where, *xxxx* is the DB2 subsystem name and *yyyy* is the data-sharing member name (the latter is needed only for data-sharing configurations). The element name is always 16 characters long, padded with blanks. Element names must be unique in the entire sysplex; therefore, to use ARM, you might run only one instance of a particular program per subsystem.

The replication programs use the element name to register with ARM during initialization. They do not provide ARM with an event exit when they register. (The event exit is not needed because the replication programs do not run as an MVS subsystem.) ARM restarts registered programs for you if they terminate abnormally (for example, if a segment violation occurs). A registered replication program de-registers if it terminates normally (for example, due to a STOP command) or if it encounters an invalid registration.

Tip: If you start the Capture or Apply program using parameter NOTERM=Y, the program does not stop when DB2 is quiesced. In this case, the program does not de-register from ARM. It continues to run but does not capture data until DB2 is restarted.

Migrating your replication environment to data-sharing mode (z/OS)

If the Capture program is running in non-data sharing mode but you migrate your installation to data-sharing mode, you must prepare your systems to run in a Sysplex by running the **ASNPLXFY** utility once. Run this utility on the data sharing configuration before warm-starting the Capture program so that the Capture program starts at the correct LRSN. This utility migrates the data in the restart (IBMSNAP_RESTART) table. It converts the non-data sharing log sequence numbers (RBA) to the equivalent sequence numbers (LRSN) in a data-sharing environment.

Prerequisites:

Use either the same user ID that you use to run the Capture program, or one that has the same privileges. Ensure that the **ASNPLXFY** utility is APF authorized. The ASNPLXFY plan must be bound to the subsystem. Also, the subsystem must be running in data sharing mode. For details about binding this utility, see the Program Directory.

Procedure:

To run the **ASNPLXFY** utility in the USS data-sharing environment:

1. Stop the Capture program.
2. Type the following command from a command line:
   ```
   ASNPLXFY yoursubsystem captureschema
   ```
   where the name of the subsystem is required and the Capture schema is optional. The default Capture schema is ASN.
3. Warm-start the Capture program.

Chapter 21. Using the Windows Service Control Manager to issue system commands for SQL replication (Windows)

This section describes how to create services that start the replication programs on Windows operating systems. You can create services for each Capture control server, Apply control server, and Monitor control server. The services are grouped with other DB2 services. If you want to change the parameters for a program after the service is started, you must drop the service and create a new one.

- "Creating a replication service"
- "Operating a replication service" on page 410
- "Dropping a replication service" on page 410

Creating a replication service

Before you create a replication service, make sure that the DB2 instance service is running. If the DB2 instance service is not running when you create the replication service, the replication service is created but it is not started automatically.

To create a replication service, use the **asnscrt** command. See "asnscrt: Creating a DB2 replication service to start the replication programs" on page 302 for command syntax and parameter descriptions.

Tip: If your replication service is set up correctly, the service name is sent to stdout after the service is started successfully. If the service does not start, check the log files for the program that you were trying to start. By default, the log files are in the directory specified by the DB2PATH environment variable. You can override this default by specifying the path parameter (**capture_path**, **apply_path**, **monitor_path**) for the program that is started as a service. Also, you can use the Windows Service Control Manager (SCM) to view the status of the service.

When you create a service, you must specify the account name that you use to log on to Windows and the password for that account name.

You can add more than one replication service to your system. You can add one service for each schema on every Capture server, and for each qualifier on every Apply control server and Monitor control server, respectively. For example, if you have five databases and each database is an Apply control server, a Capture control server, and a Monitor control server, you can create fifteen replication services. If you have multiple schemas or qualifiers on each server, you could create more services.

When you create a replication service, it is added to the SCM in Automatic mode and the service is started. Windows registers the service under a unique service name and display name.

Replication service name

The replication service name uniquely identifies each service and is what you use when you want to stop or start a service. It has the following format:

`DB2.instance.alias.program.qualifier_or_schema`

where:

- *instance* is the name of the DB2 instance.
- *alias* is the database alias of the Capture control server, Apply control server, or Monitor control server.
- *program* is one of the following values: CAP (for Capture program), APP (for Apply program), or MON (for Replication Alert Monitor program)
- *qualifier_or_schema* is one of the following identifiers: Apply qualifier, Monitor qualifier, or Capture schema

Example: The following service name is for a Capture program that has the schema ASN and is working with database DB1 under the instance called INST1:

```
DB2.INST1.DB1.CAP.ASN
```

Display name for the replication service

The display name is a text string that you see in the Services window and it is a more readable form of the service name. For example:

```
DB2 - INST1 DB1 CAPTURE ASN
```

If you want to add a description for the service, use the Service Control Manager (SCM) after you create a replication service. You can also use the SCM to specify a user name and a password for a service.

Operating a replication service

After you create a replication service, you can stop it and start it again.

Use one of the following methods to stop a service:

- SCM
- **net stop** command

Important: When you stop a replication service, the program associated with that service is stopped automatically. However, if you stop a program using a replication system command (**asnacmd**, **asnccmd**, or **asnmcmd**), the service that you used to start that program continues running until you stop it explicitly.

Use one of the following methods to start a service for replication commands:

- SCM
- **net start** command

Important: If you started a replication program from a service, you will get an error if you try to start the program using the same schema or qualifier.

Dropping a replication service

If you no longer need a replication service you can drop it so that it is removed from the SCM. Also, if you want to change the start-up parameters for a program that is started by a service, you must drop the service and then create a new one using new start-up parameters.

To drop a service for replication commands, use the **asnsdrop** command.

Related reference:

- “asnscrt: Creating a DB2 replication service to start the replication programs” on page 302
- “asnsdrop: Dropping DB2 replication services” on page 305

Chapter 22. Scheduling SQL replication programs on various operating systems

You might want to schedule the Capture program, the Apply program, or the Replication Alert Monitor program to start at a prescribed time using the commands that are available on your operating system.

Scheduling programs on UNIX and Linux operating systems

Use the **at** command to start a program at a specific time. For example, the following commands start the programs at 3:00 p.m. on Friday:
Scheduling the Capture program:

```
at 3pm Friday asncap autoprune=n
```

Scheduling the Apply program:

```
at 3pm Friday asnapply applyqual=myqual
```

Scheduling the Replication Alert Monitor program:

```
at 3pm Friday asnmon monitor_server=db2srv1  monitor_qualifier=mymon
```

Scheduling programs on Windows operating systems

If you're not using the Windows Service Control Manager, you might want to use the **AT** command to start the programs at a specific time. Before you enter the **AT** command, start the Windows Schedule Service.

The following examples start each program at 15:00 (3:00 p.m.):
Scheduling the Capture program:

```
c:\>AT 15:00 /interactive "c:\SQLLIB\BIN\db2cmd.exe c:\CAPTURE\asncap.exe"
```

Scheduling the Apply program:

```
c:\>AT 15:00 /interactive "c:\SQLLIB\BIN\db2cmd.exe
    c:\SQLLIB\BIN\asnapply.exe control_server=cntldb apply_qual=qualid1"
```

Scheduling the Replication Alert Monitor program:

```
c:\>AT 15:00 /interactive "c:\SQLLIB\BIN\db2cmd.exe
    c:\CAPTURE\asnmon.exe monitor_server=db2srv1  monitor_qualifier=mymon"
```

Scheduling programs on z/OS operating systems

Use either the **$TA JES2** command or the **AT NetView** command to start Capture for z/OS at a specific time.

To schedule a program on z/OS:

1. Create a procedure that calls the program for z/OS in the PROCLIB.
2. Modify the ICHRIN03 RACF module (or appropriate definitions for your MVS security package) to associate the procedure with a user ID.
3. Link-edit the module in SYS1.LPALIB.

See *MVS/ESA JES2 Commands* for more information about using the **$TA JES2** command, and the *NetView for MVS Command Reference* for more information about using the **AT NetView** command.

Scheduling programs on the OS/400 operating system

Use the **ADDJOBSCDE** command to start the Apply program at a specific time.

Use the **SBMJOB** command to schedule the start of the Capture program on OS/400:

```
SBMJOB CMD('STRDPRCAP...')SCDDATE(...)SCDTIME(...)
```

Chapter 23. How the SQL replication components communicate

The replication components run independently of each other, and they rely on information that they each store in the replication control tables to communicate with each other. DB2® replication has four components:

- The Replication Center
- The Capture program or triggers
- The Apply program
- The Replication Alert Monitor

The Replication Center stores the initial information about registered sources, subscription sets, and alert conditions in the control tables. The Capture program, the Apply program, and the Capture triggers update the control tables to indicate the progress of replication and to coordinate the processing of changes. The Replication Alert Monitor reads the control tables that have been updated by the Capture program, Apply program, and the Capture triggers to understand the problems and progress at a server.

The Replication Center, the Capture program or triggers, and the Apply program

When you register a table, view, or nickname as a replication source, the Replication Center creates an SQL script that stores the information for this source in the replication control table that contains all registration information, the register (IBMSNAP_REGISTER) table. The SQL script generated by the Replication Center also creates the CD tables for the registered sources.

The IBMSNAP_REGISTER table contains one row for every registered source table, and one row for every underlying table in a registered view. This table contains the following kinds of information about each registered source:

- The schema name and name of the source table
- The structure type of each registered source table
- The schema name and name of the CD table
- For registered views, the names of the CD tables for the underlying tables in this view (if the underlying tables are registered)
- The schema name and name of the internal CCD table, if there is one
- The conflict-detection level for update-anywhere sources

The Capture and Apply programs use the information in the IBMSNAP_REGISTER table to communicate their respective status to each other. This table has several more columns for related information. See the "*schema*.IBMSNAP_REGISTER" on page 449 for more information about this table.

For OS/400® sources, including tables that are journaled remotely, there is also an extension to the IBMSNAP_REGISTER table, IBMSNAP_REG_EXT, which contains extra information that is unique to iSeries™ systems, for example, the journal library and the journal name.

When you create a subscription set and add members to it, the Replication Center creates an SQL script that stores the information for this subscription set in the replication control tables that contain all subscription-set information: the subscription set (IBMSNAP_SUBS_SET), subscription-set member (IBMSNAP_SUBS_MEMBR), subscription-set columns (IBMSNAP_SUBS_COLS), and subscription-set statements (IBMSNAP_SUBS_STMTS) tables. The SQL script generated by the Replication Center also creates the target tables if they do not already exist.

The main subscription-set table, IBMSNAP_SUBS_SET, contains one row for every subscription set. This table contains the following kinds of information about each subscription set:

- The Apply qualifier
- The name of the subscription set
- The type of subscription set: read only or read/write (update anywhere)
- The names and aliases of the source and target databases
- The timing for processing the subscription set
- The current status for the subscription set

This table has several more columns for related information. See the"ASN.IBMSNAP_SUBS_SET" on page 480 for more information about this table.

The other subscription-set tables contain information about the subscription-set members, columns, and SQL statements (or stored procedures) that are processed with the set.

The Capture program and the Apply program

The Capture program uses some of the replication control tables to indicate what changes have been made to the source database, and the Apply program uses these control-table values to detect what needs to be copied to the target database. The Capture program does not capture any information until the Apply program signals it to do so, and the Apply program does not signal the Capture program to start capturing changes until you define a replication source and associated subscription sets.

The following process describes how the Apply and Capture programs communicate, in a *typical* replication scenario, to ensure data integrity:

Capturing data from a source database

1. The Capture program reads the IBMSNAP_REGISTER table during startup to determine which registered replication sources it must capture changes for, and it holds the registration information in memory.
2. The Capture program reads the DB2 log or journal continuously to detect change records (INSERT, UPDATE, and DELETE) for registered source tables or views. It also detects inserts to the signal (IBMSNAP_SIGNAL) table in order to pick up signal actions that have been initialized by the Apply program or a user. When the Apply program inserts a CAPSTART signal in the IBMSNAP_SIGNAL table and the Capture program detects the committed signal, the Capture program initializes the registration and starts capturing changes for the associated source.
3. Once the Capture program has started capturing changes for a registered source, the program writes one row (or two rows if you specified that updates should be saved as DELETE and INSERT statements) to the CD table for each

committed change that it finds in the DB2 log or journal. The Capture program keeps uncommitted changes in memory until the changes are committed or aborted. Each registered replication source that is not an external CCD table has an associated CD table.

4. At each commit interval, the Capture program commits the data that it has written to the CD and UOW tables, and also updates the IBMSNAP_REGISTER table to flag which CD tables have new committed changes.

Applying data to a target database

5. For all newly defined subscription sets, the Apply program first signals the Capture program to start capturing changes. Then a full refresh is performed for each member of the set (unless it is not a complete target table).
6. When any subscription set is eligible for replication, the Apply program checks the IBMSNAP_REGISTER table to determine whether there are changes that need to be replicated.
7. The Apply program copies the changes from the CD table to the target table.
8. The Apply program updates the IBMSNAP_SUBS_SET table to record how much data the Apply program copied for each subscription set.
9. The Apply program updates the prune set (IBMSNAP_PRUNE_SET) table with a value that indicates the point to which it has read changes from the CD table.

Pruning the CD tables

10. When the Capture program prunes the CD tables, it uses the information in the IBMSNAP_PRUNE_SET table to determine which changes were applied, and deletes these already-replicated changes from the CD table.

The Capture triggers and the Apply program

The Capture triggers use some of the replication control tables to indicate what changes have been made to the source database, and the Apply program uses these control-table values to detect what needs to be copied to the target database.

The Capture triggers start capturing information immediately. Unlike the Capture program, they do not wait for a signal from the Apply program.

The following process describes how the Capture triggers and the Apply program communicate, in a *typical* replication scenario, to ensure data integrity:

Capturing data from a source

1. Whenever a DELETE, UPDATE, or INSERT operation occurs at the registered replication source table, a Capture trigger records the change in the CCD table for that source table.

Applying data to a target

2. For all newly defined subscription sets, the Apply program first signals the Capture triggers to mark a valid starting point within the CCD table from which to start fetching changed data. Then a full refresh is performed for each member of the set (unless it is not a complete target table).
3. When the Apply program processes a subscription set for a non-DB2 relational source, it updates the register synchronization (IBMSNAP_REG_SYNCH) table, which causes an UPDATE trigger on that table to fire. The trigger updates the SYNCHPOINT value in the IBMSNAP_REGISTER table to mark the highest SYNCHPOINT value in the CCD tables that it copied to the targets. During the next cycle, the Apply program will process new data in the CCD table that has a SYNCHPOINT value that is less than or equal to this SYNCHPOINT. Because

the IBMSNAP_REG_SYNCH table is in the non-DB2 database, the Apply program writes to the table using the nickname for it that was created by the Replication Center.

4. The Apply program checks the IBMSNAP_REGISTER table to determine whether there are changes that need to be replicated.
5. The Apply program copies the changes from the CCD table to the target table.
6. The Apply program updates the subscription set (IBMSNAP_SUBS_SET) table to record how much data the Apply program copied for each subscription set.
7. The Apply program updates the prune control (IBMSNAP_PRUNCNTL) table for each registered source with a value that indicates the point to which it has read changes from the CCD table.

Pruning the CCD tables

8. The UPDATE trigger on the IBMSNAP_PRUNCNTL table checks all of the CCD tables in the source database, and deletes the already-replicated changes from the CCD table.

The Replication Center and the Replication Alert Monitor

When you define an alert condition with contacts who will be notified when the condition occurs, the Replication Center creates an SQL script that stores the information for this alert condition and its contacts in the replication control tables that contain all alert-condition and notification information: the Monitor conditions (IBMSNAP_CONDITIONS), Monitor contacts (IBMSNAP_CONTACTS), Monitor groups (IBMSNAP_GROUPS), and Monitor group contacts (IBMSNAP_CONTACTGRP) tables.

The main monitor alert table, the Monitor conditions table, contains one row for each condition that you want to be monitored. The table contains the following kinds of information about each alert condition:

- The Monitor qualifier
- The name and aliases of the Capture server or Apply server you want monitored
- The component that you want monitored (the Capture program or the Apply program)
- The Capture schema or Apply qualifier
- The name of the subscription set (if you want to monitor a set)
- The alert condition that you want monitored
- The contact to be notified if the condition occurs

This table has several more columns for related information. See "IBMSNAP_CONDITIONS table" on page 488 for more information about this table.

The other tables for the Replication Alert Monitor contain information about who will be notified if the alert condition occurs (either an individual contact or a group of contacts), how that contact will be notified (through e-mail or pager), and how often the contact will be notified if the condition continues.

The Replication Alert Monitor, the Capture program, and the Apply program

The Replication Alert Monitor uses some of the Capture control tables to monitor the Capture program, and uses some of the Apply control tables to monitor the Apply program. It reads from different replication control tables at each Capture control server or Apply control server, depending on what it is monitoring. The Replication Alert Monitor does not interfere or communicate with the Capture or Apply program.

The following process describes how the Replication Alert Monitor monitors conditions for the Capture or Apply program and notifies contacts when the alert condition occurs:

1. The Replication Alert Monitor reads the alert conditions and the contact for each condition (for a Monitor qualifier) in the Monitor conditions (IBMSNAP_CONDITIONS) table.
2. For each Capture control server or Apply control server that has a defined alert condition, the Replication Alert Monitor performs the following tasks:
   a. The Replication Alert Monitor connects to the server and reads the replication control tables associated with each alert condition for that server to see if any of the conditions are met.
   b. If any condition is met, the Replication Alert Monitor stores the data that is related to that condition in memory and continues processing the remaining alert conditions for that server.
   c. When it is finished processing all the alert conditions for that server, the Replication Alert Monitor disconnects from the Capture control or Apply control server, inserts the alerts in the Monitor alerts (IBMSNAP_ALERTS) table, and notifies the contacts for that condition.

Related concepts:

- Chapter 15, "Using the Replication Center for SQL replication," on page 219

Related reference:

- "List of tables used at the Apply control server" on page 431
- "List of tables used at the Capture control server" on page 428
- "List of control tables at the Monitor control server" on page 432

Chapter 24. Table structures for SQL replication

This chapter describes the relational database tables that are used for replication at each server: the Capture control server, Apply control server, Monitor control server, and target server. The chapter provides three ways for you to reference the tables:

- The "Tables at a glance" section provides quick reference sheets, which include the list of tables for the Capture control server, Apply control server, and Monitor control server, the columns in each table, and the indexes on each table.
- For an overview of the tables at each server, see:
  - "List of tables used at the Capture control server" on page 428
  - "List of tables used at the Apply control server" on page 431
  - "List of control tables at the Monitor control server" on page 432
  - "List of tables used at the target server" on page 433
- For a more detailed description about the tables at each server and a description of the columns in each table, see:
  - "Tables at the Capture control server and their column descriptions" on page 433
  - "Tables at the Apply control server and their column descriptions" on page 463
  - "Tables at the Monitor control server and their column descriptions" on page 487
  - "Tables at the target server and their column descriptions" on page 501.

In each section, the control tables are listed in alphabetical order by the actual table name (for example, IBMSNAP_APPLYTRACE), and the target tables are listed in alphabetical order by their English table name (for example, replica table). The columns for each table are listed in the order in which they appear in the table.

Some of the control tables require that you *not* use SQL to update them (see particular table descriptions for details). Altering control tables inappropriately can cause problems such as unexpected results, loss of data, and reduced replication performance.

Tables at a glance

Figure 15 on page 422, Figure 16 on page 423, and Figure 17 on page 424 show the tables at the Capture control server, the columns in each table, and the indexes on each table. Figure 19 on page 426 and Figure 18 on page 425 show the tables at the Apply control server, the columns in each table, and the indexes on each table. Figure 20 on page 427 and Figure 21 on page 428 show the tables at the Monitor control server, the columns in each table, and the indexes on each table.

Control tables used at the Capture control server (image 1 of 3)

***schema*.IBMSNAP_AUTHTKN**

OS/400 only
(JRN_LIB, JRN_NAME)

APPLY_QUAL	CHAR(18) NOT NULL
IBMSNAP_AUTHTKN	CHAR(26) NOT NULL
JRN_LIB	CHAR(10) NOT NULL
JRN_NAME	CHAR(10) NOT NULL
IBMSNAP_LOGMARKER	TIMESTAMP NOT NULL

***schema*.IBMSNAP_CAPENQ**

UNIX, Windows, and z/OS only
(no index)

LOCK_NAME	CHAR(9)

***schema*.IBMSNAP_CAPMON**

(MONITOR_TIME)

MONITOR_TIME	TIMESTAMP NOT NULL
RESTART_TIME	TIMESTAMP NOT NULL
CURRENT_MEMORY	INT NOT NULL
CD_ROWS_INSERTED	INT NOT NULL
RECAP_ROWS_SKIPPED	INT NOT NULL
TRIGR_ROWS_SKIPPED	INT NOT NULL
CHG_ROWS_SKIPPED	INT NOT NULL
TRANS_PROCESSED	INT NOT NULL
TRANS_SPILLED	INT NOT NULL
MAX_TRANS_SIZE	INT NOT NULL
LOCKING_RETRIES	INT NOT NULL
JRN_LIB	CHAR(10)
JRN_NAME	CHAR(10)
LOGREADLIMIT	INT NOT NULL
CAPTURE_IDLE	INT NOT NULL
SYNCHTIME	TIMESTAMP NOT NULL

ASN.IBMSNAP_CAPSCHEMAS

UNIX, Windows, and z/OS only
(CAP_SCHEMA_NAME)

CAP_SCHEMA_NAME	VARCHAR(30)[1]

OS/400 only
(CAP_SCHEMA_NAME)

CAP_SCHEMA_NAME	VARCHAR(30)
STATUS	CHAR(1)

[1] VARCHAR(30) for DB2 for z/OS V8 compatibility mode or earlier; VARCHAR(128) for DB2 for z/OS V8 new-function mode.

***schema*.IBMSNAP_CAPPARMS**

(no index)

RETENTION_LIMIT	INT
LAG_LIMIT	INT
COMMIT_INTERVAL	INT
PRUNE_INTERVAL	INT
TRACE_LIMIT	INT
MONITOR_LIMIT	INT
MONITOR_INTERVAL	INT
MEMORY_LIMIT	SMALLINT
REMOTE_SRC_SERVER	CHAR(18)
AUTOPRUNE	CHAR(1)
TERM	CHAR(1)
AUTOSTOP	CHAR(1)
LOGREUSE	CHAR(1)
LOGSTDOUT	CHAR(1)
SLEEP_INTERVAL	SMALLINT
CAPTURE_PATH	VARCHAR(1040)
STARTMODE	VARCHAR(10)

***schema*.IBMSNAP_CAPTRACE**

UNIX, Windows, and z/OS only
(TRACE_TIME)

OPERATION	CHAR(8) NOT NULL
TRACE_TIME	TIMESTAMP NOT NULL
DESCRIPTION	VARCHAR(1024) NOT NULL

OS/400
(TRACE_TIME)

OPERATION	CHAR(8) NOT NULL
TRACE_TIME	TIMESTAMP NOT NULL
JOB_NAME	CHAR(26) NOT NULL
JOB_STR_TIME	TIMESTAMP NOT NULL
DESCRIPTION	VARCHAR(298) NOT NULL

***schema*.IBMSNAP_PARTITIONINFO**

(PARTITIONID, USAGE)

PARTITIONID	INT NOT NULL
USAGE	CHAR(1)
SEQUENCE	CHAR(10) FOR BIT DATA NOT NULL
STATUS	CHAR(1)
LAST_UPDATE	TIMESTAMP

***schema*.IBMSNAP_PRUNE_LOCK**

(no index)

DUMMY	CHAR(1)

Figure 15. Tables used at the Capture control server. These tables are used by the Capture program, Apply program, and Capture triggers at the Capture control server. The columns that make up each table's main index are listed in parentheses under the table name.

Control tables used at the Capture control server (image 2 of 3)

***schema*.IBMSNAP_PRUNE_SET**

(TARGET_SERVER, APPLY_QUAL, SET_NAME)

TARGET_SERVER	CHAR(18) NOT NULL
APPLY_QUAL	CHAR(18) NOT NULL
SET_NAME	CHAR(18) NOT NULL
SYNCHTIME	TIMESTAMP
SYNCHPOINT	CHAR(10) FOR BIT DATA NOT NULL

***schema*.IBMSNAP_PRUNCNTL**

(SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL, APPLY_QUAL, SET_NAME, TARGET_SERVER, TARGET_TABLE, TARGET_OWNER)

TARGET_SERVER	CHAR(18) NOT NULL
TARGET_OWNER	VARCHAR(30)[1] NOT NULL
TARGET_TABLE	VARCHAR(128)[2] NOT NULL
SYNCHTIME	TIMESTAMP
SYNCHPOINT	CHAR(10) FOR BIT DATA
SOURCE_OWNER	VARCHAR(30)[1] NOT NULL
SOURCE_TABLE	VARCHAR(128)[2] NOT NULL
SOURCE_VIEW_QUAL	SMALLINT NOT NULL
APPLY_QUAL	CHAR(18) NOT NULL
SET_NAME	CHAR(18) NOT NULL
CNTL_SERVER	CHAR(18) NOT NULL
TARGET_STRUCTURE	SMALLINT NOT NULL
CNTL_ALIAS	CHAR(8)
PHYS_CHANGE_OWNER	VARCHAR(30)[1]
PHYS_CHANGE_TABLE	VARCHAR(128)[2]
MAP_ID	VARCHAR(10) NOT NULL

***schema*.IBMSNAP_REG_EXT**

OS/400 only

(VERSION, SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL)

VERSION	INT NOT NULL
SOURCE _OWNER	VARCHAR(30) NOT NULL
SOURCE_TABLE	VARCHAR(128) NOT NULL
SOURCE_NAME	CHAR(10)
SOURCE_MBR	CHAR(10)
SOURCE_TABLE_RDB	CHAR(18)
JRN_LIB	CHAR(10)
JRN_NAME	CHAR(10)
FR_START_TIME	TIMESTAMP
SOURCE_VIEW_QUAL	SMALLINT NOT NULL
CMT_BEHAVIOR_CASE	SMALLINT NOT NULL WITH DEFAULT
MAX_ROWS_BTWN_CMTS	SMALLINT NOT NULL WITH DEFAULT

***schema*.IBMSNAP_REGISTER**

(SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL)

SOURCE _OWNER	VARCHAR(30)[1] NOT NULL
SOURCE_TABLE	VARCHAR(128)[2] NOT NULL
SOURCE_VIEW_QUAL	SMALLINT NOT NULL
GLOBAL_RECORD	CHAR(1) NOT NULL
SOURCE_STRUCTURE	SMALLINT NOT NULL
SOURCE_CONDENSED	CHAR(1) NOT NULL
SOURCE_COMPLETE	CHAR(1) NOT NULL
CD_OWNER	VARCHAR(30)[1]
CD_TABLE	VARCHAR(128)[2]
PHYS_CHANGE_OWNER	VARCHAR(30)[1]
PHYS_CHANGE_TABLE	VARCHAR(128)[2]
CD_OLD_SYNCHPOINT	CHAR(10) FOR BIT DATA
CD_NEW_SYNCHPOINT	CHAR(10) FOR BIT DATA
DISABLE_REFRESH	SMALLINT NOT NULL
CCD_OWNER	VARCHAR(30)[1]
CCD_TABLE	VARCHAR(128)[2]
CCD_OLD_SYNCHPOINT	CHAR(10) FOR BIT DATA
SYNCHPOINT	CHAR(10) FOR BIT DATA
SYNCHTIME	TIMESTAMP
CCD_CONDENSED	CHAR(1)
CCD_COMPLETE	CHAR(1)
ARCH_LEVEL	CHAR(4) NOT NULL
DESCRIPTION	CHAR(254)
BEFORE_IMG_PREFIX	VARCHAR(4)
CONFLICT_LEVEL	CHAR(1)
CHG_UPD_TO_DEL_INS	CHAR(1)
CHGONLY	CHAR(1)
RECAPTURE	CHAR(1)
OPTION_FLAGS	CHAR(4) NOT NULL
STOP_ON_ERROR	CHAR(1) WITH DEFAULT
STATE	CHAR(1) WITH DEFAULT
STATE_INFO	CHAR(8)

***schema*.IBMSNAP_REG_SYNCH**

non-DB2 relational only

(TRIGGER_ME)

TRIGGER_ME	CHAR(1) NOT NULL

[1] VARCHAR(30) for DB2 for z/OS V8 compatibility mode or earlier; VARCHAR(128) for DB2 for z/OS V8 new-function mode.

[2] VARCHAR(18) for DB2 for z/OS V8 compatibility mode or earlier; VARCHAR(128) for DB2 for z/OS V8 new-function mode.

Figure 16. Tables used at the Capture control server (continued). These tables are used by the Capture program, Apply program, and Capture triggers at the Capture control server. The columns that make up each table's main index are listed in parentheses under the table name.

Control tables used at the Capture control server (image 3 of 3)

***schema*.IBMSNAP_RESTART**

UNIX, Windows, and z/OS only
(no index)

MAX_COMMITSEQ	CHAR(10) FOR BIT DATA NOT NULL
MAX_COMMIT_TIME	TIMESTAMP NOT NULL
MIN_INFLIGHTSEQ	CHAR(10) FOR BIT DATA NOT NULL
CURR_COMMIT_TIME	TIMESTAMP NOT NULL
CAPTURE_FIRST_SEQ	CHAR(10) FOR BIT DATA NOT NULL

OS/400 only
(JRN_LIB, JRN_NAME)

MAX_COMMITSEQ	CHAR(10) FOR BIT DATA NOT NULL
MAX_COMMIT_TIME	TIMESTAMP NOT NULL
MIN_INFLIGHTSEQ	CHAR(10) FOR BIT DATA NOT NULL
CURR_COMMIT_TIME	TIMESTAMP NOT NULL
CAPTURE_FIRST_SEQ	CHAR(10) FOR BIT DATA NOT NULL
UID	INTEGER NOT NULL
SEQNBR	BIGINT NOT NULL
JRN_LIB	CHAR(10) NOT NULL
JRN_NAME	CHAR(10) NOT NULL
STATUS	CHAR(1)

***schema*.IBMSNAP_SEQTABLE**

Informix only
(SEQ)

SEQ	INTEGER NOT NULL

***schema*.IBMSNAP_SIGNAL**

(SIGNAL_TIME)

SIGNAL_TIME	TIMESTAMP NOT NULL WITH DEFAULT
SIGNAL_TYPE	VARCHAR(30) NOT NULL
SIGNAL_SUBTYPE	VARCHAR(30)
SIGNAL_INPUT_IN	VARCHAR(500)
SIGNAL_STATE	CHAR(1) NOT NULL
SIGNAL_LSN	CHAR(10) FOR BIT DATA

***schema*.IBMSNAP_UOW**

(IBMSNAP_COMMITSEQ, IBMSNAP_LOGMARKER)

IBMSNAP_UOWID	CHAR(10) FOR BIT DATA NOT NULL
IBMSNAP_COMMITSEQ	CHAR(10) FOR BIT DATA NOT NULL
IBMSNAP_LOGMARKER	TIMESTAMP NOT NULL
IBMSNAP_AUTHTKN	VARCHAR(30) NOT NULL
IBMSNAP_AUTHID	VARCHAR(30)[1] NOT NULL
IBMSNAP_REJ_CODE	CHAR(1) NOT NULL WITH DEFAULT
IBMSNAP_APPLY_QUAL	CHAR(18) NOT NULL WITH DEFAULT

[1] VARCHAR(30) for DB2 for z/OS V8 compatibility mode or earlier; VARCHAR(128) for DB2 for z/OS V8 new-function mode.

Figure 17. Tables used at the Capture control server (continued). These tables are used by the Capture program, Apply program, and Capture triggers at the Capture control server. The columns that make up each table's main index are listed in parentheses under the table name.

Control tables used at the Apply control server (image 1 of 2)

ASN.IBMSNAP_APPLYTRAIL

(LASTRUN, APPLY_QUAL)

APPLY_QUAL	CHAR(18) NOT NULL
SET_NAME	CHAR(18) NOT NULL
SET_TYPE	CHAR(1) NOT NULL
WHOS_ON_FIRST	CHAR(1) NOT NULL
ASNLOAD	CHAR(1)
FULL_REFRESH	CHAR(1)
EFFECTIVE_MEMBERS	INT
SET_INSERTED	INT NOT NULL
SET_DELETED	INT NOT NULL
SET_UPDATED	INT NOT NULL
SET_REWORKED	INT NOT NULL
SET_REJECTED_TRXS	INT NOT NULL
STATUS	SMALLINT NOT NULL
LASTRUN	TIMESTAMP NOT NULL
LASTSUCCESS	TIMESTAMP
SYNCHPOINT	CHAR(10) FOR BIT DATA
SYNCHTIME	TIMESTAMP
SOURCE_SERVER	CHAR (18) NOT NULL
SOURCE_ALIAS	CHAR(8)
SOURCE_OWNER	VARCHAR(30)[1]
SOURCE_TABLE	VARCHAR(128)2
SOURCE_VIEW_QUAL	SMALLINT
TARGET_SERVER	CHAR(18) NOT NULL
TARGET_ALIAS	CHAR(8)
TARGET_OWNER	VARCHAR(30)[1] NOT NULL
TARGET_TABLE	VARCHAR(128)[2] NOT NULL
CAPTURE_SCHEMA	VARCHAR(30)[1] NOT NULL
TGT_CAPTURE_SCHEMA	VARCHAR(30)[1]
FEDERATED_SRC_SRVR	VARCHAR(18)
FEDERATED_TGT_SRVR	VARCHAR(18)
JRN_LIB	CHAR(10)
JRN_NAME	CHAR(10)
COMMIT_COUNT	SMALLINT
OPTION_FLAGS	CHAR(4) NOT NULL
EVENT_NAME	CHAR(18)
ENDTIME	TIMESTAMP NOT NULL WITH DEFAULT
SOURCE_CONN_TIME	TIMESTAMP
SQLSTATE	CHAR(5)
SQLCODE	INT
SQLERRP	CHAR(8)
SQLERRM	VARCHAR(70)
APPERRM	VARCHAR(760)

ASN.IBMSNAP_APPENQ

(APPLY_QUAL)

APPLY_QUAL	CHAR(18)

ASN.IBMSNAP_APPLY_JOB

OS/400 only
(no index)

APPLY_QUAL	CHAR(18) NOT NULL
CONTROL_SERVER	CHAR(18) NOT NULL
JOB_NAME	CHAR(10) NOT NULL
USER_NAME	CHAR(10) NOT NULL
JOB_NUMBER	CHAR(6) NOT NULL

ASN.IBMSNAP_APPLYTRACE

(APPLY_QUAL, TRACE_TIME)

APPLY_QUAL	CHAR(18) NOT NULL
TRACE_TIME	TIMESTAMP NOT NULL
OPERATION	CHAR(8) NOT NULL
DESCRIPTION	VARCHAR(1024) NOT NULL

ASN.IBMSNAP_APPPARMS

(APPLY_QUAL)

APPLY_QUAL	CHAR(18) NOT NULL
APPLY_PATH	VARCHAR(1040)
COPYONCE	CHAR(1) WITH DEFAULT
DELAY	INT WITH DEFAULT
ERRWAIT	INT WITH DEFAULT
INAMSG	CHAR(1) WITH DEFAULT
LOADXIT	CHAR(1) WITH DEFAULT
LOGREUSE	CHAR(1) WITH DEFAULT
LOGSTDOUT	CHAR(1) WITH DEFAULT
NOTIFY	CHAR(1) WITH DEFAULT
OPT4ONE	CHAR(1) WITH DEFAULT
SLEEP	CHAR(1) WITH DEFAULT
SQLERRCONTINUE	CHAR(1) WITH DEFAULT
SPILLFILE	VARCHAR(10) WITH DEFAULT
TERM	CHAR(1) WITH DEFAULT
TRLREUSE	CHAR(1) WITH DEFAULT

[1] VARCHAR(30) for DB2 for z/OS V8 compatibility mode or earlier; VARCHAR(128) for DB2 for z/OS V8 new-function mode.

[2] VARCHAR(18) for DB2 for z/OS V8 compatibility mode or earlier; VARCHAR(128) for DB2 for z/OS V8 new-function mode.

Figure 18. Tables used at the Apply control server. These tables are used by the Apply program at the Apply control server. The columns that make up each table's main index are listed in parentheses under the table name.

Control tables used at the Apply control server (image 2 of 2)

ASN.IBMSNAP_SUBS_COLS

(APPLY_QUAL, SET_NAME, WHOS_ON_FIRST, TARGET_OWNER, TARGET_TABLE, TARGET_NAME)

APPLY_QUAL	CHAR(18) NOT NULL
SET_NAME	CHAR(18) NOT NULL
WHOS_ON_FIRST	CHAR(1) NOT NULL
TARGET_OWNER	VARCHAR(30)[1] NOT NULL
TARGET_TABLE	VARCHAR(128)[2] NOT NULL
COL_TYPE	CHAR(1) NOT NULL
TARGET_NAME	VARCHAR(30) NOT NULL
IS_KEY	CHAR(1) NOT NULL
COLNO	SMALLINT NOT NULL
EXPRESSION	VARCHAR(254) NOT NULL

ASN.IBMSNAP_SUBS_EVENT

(EVENT_NAME, EVENT_TIME)

EVENT_NAME	CHAR(18) NOT NULL
EVENT_TIME	TIMESTAMP NOT NULL
END_SYNCHPOINT	CHAR(10) FOR BIT DATA
END_OF_PERIOD	TIMESTAMP

ASN.IBMSNAP_SUBS_MEMBR

(APPLY_QUAL, SET_NAME, WHOS_ON_FIRST, SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL, TARGET_OWNER, TARGET_TABLE)

APPLY_QUAL	CHAR(18) NOT NULL
SET_NAME	CHAR(18) NOT NULL
WHOS_ON_FIRST	CHAR(1) NOT NULL
SOURCE_OWNER	VARCHAR(30)[1] NOT NULL
SOURCE_TABLE	VARCHAR(128)[2] NOT NULL
SOURCE_VIEW_QUAL	SMALLINT NOT NULL
TARGET_OWNER	VARCHAR(30)[1] NOT NULL
TARGET_TABLE	VARCHAR(128)[2] NOT NULL
TARGET_CONDENSED	CHAR(1) NOT NULL
TARGET_COMPLETE	CHAR(1) NOT NULL
TARGET_STRUCTURE	SMALLINT NOT NULL
PREDICATES	VARCHAR(1024)
MEMBER_STATE	CHAR(1)
TARGET_KEY_CHG	CHAR(1) NOT NULL
UOW_CD_PREDICATES	VARCHAR(1024)
JOIN_UOW_CD	CHAR(1)
LOADX_TYPE	SMALLINT
LOADX_SRC_N_OWNER	VARCHAR(30)[1]
LOADX_SRC_N_TABLE	VARCHAR(128)[2]

ASN.IBMSNAP_SUBS_SET

(APPLY_QUAL, SET_NAME, WHOS_ON_FIRST)

APPLY_QUAL	CHAR(18) NOT NULL
SET_NAME	CHAR(18) NOT NULL
SET_TYPE	CHAR(1) NOT NULL
WHOS_ON_FIRST	CHAR(1) NOT NULL
ACTIVATE	SMALLINT NOT NULL
SOURCE_SERVER	CHAR(18) NOT NULL
SOURCE_ALIAS	CHAR(8)
TARGET_SERVER	CHAR(18) NOT NULL
TARGET_ALIAS	CHAR(8)
STATUS	SMALLINT NOT NULL
LASTRUN	TIMESTAMP NOT NULL
REFRESH_TYPE	CHAR(1) NOT NULL
SLEEP_MINUTES	INT
EVENT_NAME	CHAR(18)
LASTSUCCESS	TIMESTAMP
SYNCHPOINT	CHAR(10) FOR BIT DATA
SYNCHTIME	TIMESTAMP
CAPTURE_SCHEMA	VARCHAR(30)[1] NOT NULL
TGT_CAPTURE_SCHEMA	VARCHAR(30)[1]
FEDERATED_SRC_SRVR	VARCHAR(18)
FEDERATED_TGT_SRVR	VARCHAR(18)
JRN_LIB	CHAR(10)
JRN_NAME	CHAR(10)
OPTION_FLAGS	CHAR(4) NOT NULL
COMMIT_COUNT	SMALLINT
MAX_SYNCH_MINUTES	SMALLINT
AUX_STMTS	SMALLINT NOT NULL
ARCH_LEVEL	CHAR(4) NOT NULL

ASN.IBMSNAP_SUBS_STMTS

(APPLY_QUAL, SET_NAME, WHOS_ON_FIRST, BEFORE_OR_AFTER, STMT_NUMBER)

APPLY_QUAL	CHAR(18) NOT NULL
SET_NAME	CHAR(18) NOT NULL
WHOS_ON_FIRST	CHAR(1) NOT NULL
BEFORE_OR_AFTER	CHAR(1) NOT NULL
STMT_NUMBER	SMALLINT NOT NULL
EI_OR_CALL	CHAR(1) NOT NULL
SQL_STMT	VARCHAR(1024)
ACCEPT_SQLSTATES	VARCHAR(50)

[1] VARCHAR(30) for DB2 for z/OS V8 compatibility mode or earlier; VARCHAR(128) for DB2 for z/OS V8 new-function mode.

[2] VARCHAR(18) for DB2 for z/OS V8 compatibility mode or earlier; VARCHAR(128) for DB2 for z/OS V8 new-function mode.

Figure 19. Tables used at the Apply control server (continued). These tables are used by the Apply program at the Apply control server. The columns that make up each table's main index are listed in parentheses under the table name.

Control tables used at the Monitor control server (image 1 of 2)

ASN.IBMSNAP_ALERTS

(MONITOR_QUAL, COMPONENT, SERVER_NAME, SCHEMA_OR_QUAL, SET_NAME, CONDITION_NAME, ALERT_CODE)

MONITOR_QUAL	CHAR(18) NOT NULL
ALERT_TIME	TIMESTAMP NOT NULL
COMPONENT	CHAR(1) NOT NULL
SERVER_NAME	CHAR(18) NOT NULL
SERVER_ALIAS	CHAR(8)
SCHEMA_OR_QUAL	VARCHAR(30)[1] NOT NULL
SET_NAME	CHAR(18) NOT NULL WITH DEFAULT
CONDITION_NAME	CHAR(18) NOT NULL
OCCURRED_TIME	TIMESTAMP NOT NULL
ALERT_COUNTER	SMALLINT NOT NULL
ALERT_CODE	CHAR(10) NOT NULL
RETURN_CODE	INT NOT NULL
NOTIFICATION_SENT	CHAR(1) NOT NULL
ALERT_MESSAGE	VARCHAR(1024) NOT NULL

ASN.IBMSNAP_CONDITIONS

(MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL, SET_NAME, CONDITION_NAME)

MONITOR_QUAL	CHAR(18) NOT NULL
SERVER_NAME	CHAR(18) NOT NULL
COMPONENT	CHAR(1) NOT NULL
SCHEMA_OR_QUAL	VARCHAR(30)[1] NOT NULL
SET_NAME	CHAR(18) NOT NULL WITH DEFAULT
SERVER_ALIAS	CHAR(8)
ENABLED	CHAR(1) NOT NULL
CONDITION_NAME	CHAR(18) NOT NULL
PARM_INT	INT
PARM_CHAR	VARCHAR(128)
CONTACT_TYPE	CHAR(1) NOT NULL
CONTACT	VARCHAR(127) NOT NULL

ASN.IBMSNAP_CONTACTGRP

(GROUP_NAME, CONTACT_NAME)

GROUP_NAME	VARCHAR(127) NOT NULL
CONTACT_NAME	VARCHAR(127) NOT NULL

ASN.IBMSNAP_CONTACTS

(CONTACT_NAME)

CONTACT_NAME	VARCHAR(127) NOT NULL
EMAIL_ADDRESS	VARCHAR(128) NOT NULL
ADDRESS_TYPE	CHAR(1) NOT NULL
DELEGATE	VARCHAR(127)
DELEGATE_START	DATE
DELEGATE_END	DATE
DESCRIPTION	VARCHAR(1024)

ASN.IBMSNAP_GROUPS

(GROUP_NAME)

GROUP_NAME	VARCHAR(127) NOT NULL
DESCRIPTION	VARCHAR(1024)

ASN.IBMSNAP_MONENQ

(no index)

MONITOR_QUAL	CHAR(18) NOT NULL

ASN.IBMSNAP_MONPARMS

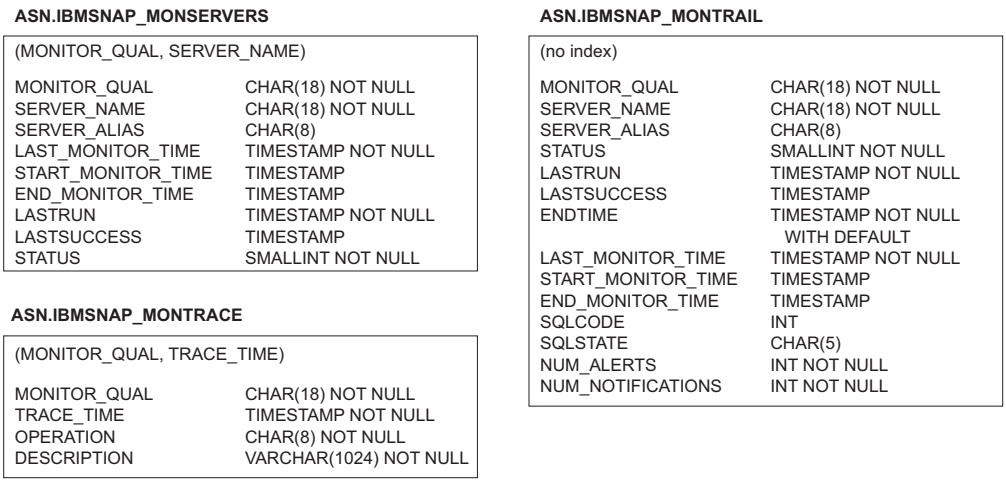
(MONITOR_QUAL)

MONITOR_QUAL	CHAR(18) NOT NULL
ALERT_PRUNE_LIMIT	INT WITH DEFAULT
AUTOPRUNE	CHAR(1) WITH DEFAULT
EMAIL_SERVER	VARCHAR(128)
LOGREUSE	CHAR(1) WITH DEFAULT
LOGSTDOUT	CHAR(1) WITH DEFAULT
NOTIF_PER_ALERT	INT WITH DEFAULT
NOTIF_MINUTES	INT WITH DEFAULT
MONITOR_ERRORS	VARCHAR(128)
MONITOR_INTERVAL	INT WITH DEFAULT
MONITOR_PATH	VARCHAR(1040)
RUNONCE	CHAR(1) WITH DEFAULT
TERM	CHAR(1)WITH DEFAULT
TRACE_LIMIT	INT WITH DEFAULT

[1] VARCHAR(30) for DB2 for z/OS V8 compatibility mode or earlier; VARCHAR(128) for DB2 for z/OS V8 new-function mode.

Figure 20. Tables used at the Monitor control server. These tables are used by the Replication Alert Monitor program at the Monitor control server. The columns that make up each table's main index are listed in parentheses under the table name.

Control tables used at the Monitor control server (image 2 of 2)



ASN.IBMSNAP_MONSERVERS

(MONITOR_QUAL, SERVER_NAME)

MONITOR_QUAL	CHAR(18) NOT NULL
SERVER_NAME	CHAR(18) NOT NULL
SERVER_ALIAS	CHAR(8)
LAST_MONITOR_TIME	TIMESTAMP NOT NULL
START_MONITOR_TIME	TIMESTAMP
END_MONITOR_TIME	TIMESTAMP
LASTRUN	TIMESTAMP NOT NULL
LASTSUCCESS	TIMESTAMP
STATUS	SMALLINT NOT NULL

ASN.IBMSNAP_MONTRACE

(MONITOR_QUAL, TRACE_TIME)

MONITOR_QUAL	CHAR(18) NOT NULL
TRACE_TIME	TIMESTAMP NOT NULL
OPERATION	CHAR(8) NOT NULL
DESCRIPTION	VARCHAR(1024) NOT NULL

ASN.IBMSNAP_MONTRAIL

(no index)

MONITOR_QUAL	CHAR(18) NOT NULL
SERVER_NAME	CHAR(18) NOT NULL
SERVER_ALIAS	CHAR(8)
STATUS	SMALLINT NOT NULL
LASTRUN	TIMESTAMP NOT NULL
LASTSUCCESS	TIMESTAMP
ENDTIME	TIMESTAMP NOT NULL WITH DEFAULT
LAST_MONITOR_TIME	TIMESTAMP NOT NULL
START_MONITOR_TIME	TIMESTAMP
END_MONITOR_TIME	TIMESTAMP
SQLCODE	INT
SQLSTATE	CHAR(5)
NUM_ALERTS	INT NOT NULL
NUM_NOTIFICATIONS	INT NOT NULL

Figure 21. Tables used at the Monitor control server (continued). These tables are used by the Replication Alert Monitor program at the Monitor control server. The columns that make up each table's main index are listed in parentheses under the table name.

List of tables used at the Capture control server

The tables stored at the Capture control server contain information about your registered sources and how the Capture program or triggers process the sources. For Linux, UNIX, Windows, and z/OS, you build these control tables to your specifications using the Replication Center. For OS/400, these control tables are created automatically for you in the ASN library when you install DataPropagator for iSeries. You can use the system commands for replication on OS/400 to create Capture control tables in alternate capture schemas.

Table 63. Quick reference for tables used at the Capture control server

Table name	Description	See page
IBMSNAP_CAPSCHEMAS	**Capture schemas table** Contains the names of all Capture schemas.	434
schema.IBMSNAP_AUTHTKN (OS/400)	**Apply-qualifier cross-reference table** Contains information to support update-anywhere.	434
schema.IBMSNAP_CAPENQ (Linux, UNIX, Windows, z/OS)	**Capture enqueue table** For each Capture schema, this table is used to ensure that: • For DB2 for Linux, UNIX and Windows, only one Capture program is running per database. • For non-data-sharing DB2 for z/OS, only one Capture program is running per subsystem. • For data-sharing DB2 for z/OS, only one Capture program is running per data-sharing group.	435
schema.CD_table	**Change-data (CD) table** Contains information about changes that occur at the source. This table is not created until you register a replication source.	442

Table 63. Quick reference for tables used at the Capture control server (continued)

Table name	Description	See page
schema.CCD_table	**Consistent-change-data (CCD) table** Contains information about changes that occur at the source and additional columns to identify the sequential ordering of those changes.	441
schema.IBMSNAP_CAPMON	**Capture monitor table** Contains operational statistics that help monitor the progress of the Capture program.	436
schema.IBMSNAP_CAPPARMS	**Capture parameters table** Contains parameters that you can specify to control the operations of the Capture program.	437
schema.IBMSNAP_CAPTRACE	**Capture trace table** Contains important messages from the Capture program.	440
schema.IBMSNAP_PARTITIONINFO	**Partition information table** Contains information that enables the Capture program to restart from the earliest required log sequence number.	443
schema.IBMSNAP_PRUNE_LOCK	**Prune lock table** Used to serialize the Capture program's access of CD tables during a cold start or during retention-limit pruning (pruning when the retention limit is reached or exceeded).	446
schema.IBMSNAP_PRUNE_SET	**Prune set table** Coordinates the pruning of CD tables.	447
schema.IBMSNAP_PRUNCNTL	**Pruning control table** Coordinates synchpoint updates between the Capture and Apply programs.	444
schema.IBMSNAP_REG_EXT (OS/400)	**Register extension table** An extension of the register table. Contains additional information about replication sources, such as the journal name and the remote source table's database entry name.	447
schema.IBMSNAP_REGISTER	**Register table** Contains information about replication sources, such as the names of the replication source tables, their attributes, and their corresponding CD and CCD table names.	449
schema.IBMSNAP_REG_SYNCH (non-DB2 relational)	**Register synchronization table** Used when replicating from a non-DB2 relational data source. An update trigger on this table simulates the Capture program by initiating an update of the SYNCHPOINT value for all the rows in the register table before the Apply program reads the information from the register table.	455

Table 63. Quick reference for tables used at the Capture control server (continued)

Table name	Description	See page
schema.IBMSNAP_RESTART	**Restart table** Contains information that enables the Capture program to resume capturing from the correct point in the log or journal. For OS/400 environments, this table is also used to determine the starting time of the **RCVJRNE** (Receive Journal Entry) command.	456
schema.IBMSNAP_SEQTABLE (Informix)	**Sequencing table** Contains a sequence of unique numbers that DB2 replication uses as the equivalent of log sequence numbers for Informix tables.	458
schema.IBMSNAP_SIGNAL	**Signal table** Contains all signals used to prompt the Capture program. These signals can be sent manually or by the Apply program.	458
schema.IBMSNAP_UOW	**Unit-of-work (UOW) table** Provides additional information about transactions that have been committed to a source table.	461

Related reference:

- "ASN.IBMSNAP_CAPSCHEMAS" on page 434
- "*schema*.IBMSNAP_AUTHTKN (OS/400)" on page 434
- "*schema*.IBMSNAP_CAPENQ (UNIX, Windows, z/OS)" on page 435
- "*schema*.IBMSNAP_CAPMON" on page 436
- "*schema*.IBMSNAP_CAPPARMS" on page 437
- "*schema*.IBMSNAP_CAPTRACE (DB2 only)" on page 440
- "*schema.CCD_table* (non-DB2)" on page 441
- "*schema.CD_table*" on page 442
- "*schema*.IBMSNAP_PRUNCNTL" on page 444
- "*schema*.IBMSNAP_PRUNE_LOCK" on page 446
- "*schema*.IBMSNAP_PRUNE_SET" on page 447
- "*schema*.IBMSNAP_REG_EXT (OS/400)" on page 447
- "*schema*.IBMSNAP_REGISTER" on page 449
- "*schema*.IBMSNAP_REG_SYNCH (non-DB2 relational)" on page 455
- "*schema*.IBMSNAP_RESTART" on page 456
- "*schema*.IBMSNAP_SEQTABLE (Informix)" on page 458
- "*schema*.IBMSNAP_SIGNAL" on page 458
- "*schema*.IBMSNAP_UOW" on page 461
- "*schema*.IBMSNAP_PARTITIONINFO" on page 443

List of tables used at the Apply control server

The tables stored at the Apply control server contain information about your subscription definitions. For Linux, UNIX, Windows, and z/OS, you build these control tables to your specifications using the Replication Center. For OS/400, these control tables are created automatically for you when install DataPropagator for iSeries.

Table 64. Quick reference for tables used at the Apply control server

Table name	Description	See page
IBMSNAP_APPENQ	**Apply enqueue table** Used to ensure that only one Apply program is running per Apply qualifier.	463
IBMSNAP_APPLY_JOB (OS/400)	**Apply job table** Used to ensure that there is a unique Apply qualifier for each instance of the Apply program running at an Apply control server.	464
IBMSNAP_APPLYTRACE	**Apply trace table** Contains important messages from the Apply program.	467
IBMSNAP_APPLYTRAIL	**Apply trail table** Contains audit-trail information about the Apply program.	468
IBMSNAP_APPPARMS	**Apply parameters table** Contains parameters that you can modify to control the operations of the Apply program.	465
IBMSNAP_SUBS_COLS	**Subscription columns table** Maps columns in the target table or view to the corresponding columns in the source table or view.	473
IBMSNAP_SUBS_EVENT	**Subscription events table** Contains events that you define to control when the Apply program processes a subscription set.	475
IBMSNAP_SUBS_MEMBR	**Subscription members table** Identifies a source and target table pair and specifies processing information for that pair.	475
IBMSNAP_SUBS_SET	**Subscription sets table** Contains processing information for each set of subscription-set members that the Apply program processes as a group.	480
IBMSNAP_SUBS_STMTS	**Subscription statements table** Contains SQL statements or stored procedure calls that you define for a subscription set. They are invoked before or after the Apply program processes the set.	485

Related reference:

- "ASN.IBMSNAP_APPENQ" on page 463
- "ASN.IBMSNAP_APPLY_JOB (OS/400)" on page 464
- "ASN.IBMSNAP_APPLYTRACE" on page 467

- "ASN.IBMSNAP_APPLYTRAIL" on page 468
- "ASN.IBMSNAP_SUBS_COLS" on page 473
- "ASN.IBMSNAP_SUBS_EVENT" on page 475
- "ASN.IBMSNAP_SUBS_MEMBR" on page 475
- "ASN.IBMSNAP_SUBS_SET" on page 480
- "ASN.IBMSNAP_SUBS_STMTS" on page 485
- "ASN.IBMSNAP_APPPARMS" on page 465

List of control tables at the Monitor control server

The control tables at the Monitor control server contain information about when, how, and whom you want the Replication Alert Monitor to contact when an alert condition occurs. For Linux, UNIX, Windows, and z/OS, you build these control tables to your specifications using the Replication Center. DataPropagator for iSeries does not have Monitor control tables.

Table 65. Control tables at the Monitor control server

Table name	Description
IBMSNAP_ALERTS	Contains a record of all the alerts issued by the Replication Alert Monitor.
IBMSNAP_CONDITIONS	Contains the alert conditions for which the Replication Alert Monitor will contact someone, and contains the group or individual's name to contact if a particular condition occurs.
IBMSNAP_CONTACTGRP	Contains the individual contacts that make up the contact groups.
IBMSNAP_CONTACTS	Contains information on how the Replication Alert Monitor notifies each person or group when an alert condition that is associated with that contact name occurs.
IBMSNAP_GROUPS	Contains the name and description of each contact group.
IBMSNAP_MONENQ	Used to ensure that only one Replication Alert Monitor program is running per Monitor qualifier.
IBMSNAP_MONPARMS	Contains parameters that you can modify to control the operations of the Monitor program.
IBMSNAP_MONSERVERS	Contains the latest time that a server was monitored by a Replication Alert Monitor program (identified by a Monitor qualifier).
IBMSNAP_MONTRACE	Contains important messages from the Monitor program.
IBMSNAP_MONTRAIL	Contains important information about each monitor cycle.

Related reference:

- "IBMSNAP_ALERTS table" on page 487
- "IBMSNAP_CONDITIONS table" on page 488
- "IBMSNAP_CONTACTGRP table" on page 493
- "IBMSNAP_MONTRAIL table" on page 499
- "IBMSNAP_CONTACTS table" on page 494

- "IBMSNAP_GROUPS table" on page 495
- "IBMSNAP_MONENQ table" on page 495
- "IBMSNAP_MONSERVERS table" on page 497
- "IBMSNAP_MONTRACE table" on page 498
- "IBMSNAP_MONPARMS table" on page 495

List of tables used at the target server

Various types of target tables are stored at the target server. If you do not use an existing table as your target table, the Replication Center builds the target table to your specifications based on how you define the subscription-set member.

Table 66. Quick reference for target tables

Table name	Description	See page
schema.base_aggregate	**Base aggregate table** Contains data that has been aggregated from a source table.	501
schema.change_aggregate	**Change aggregate table** Contains data that has been aggregated from a CD table.	501
schema.CCD	**Consistent-change data (CCD) table** Contains information about changes that occur at the source and contains additional columns to identify the sequential ordering of those changes.	502
schema.point_in_time	**Point-in-time table** A copy of the source data, with an additional column that records the specific time in the source log that the data was committed.	504
schema.replica	**Replica table** A type of target table used for update-anywhere replication.	505
schema.user_copy	**User copy table** A copy of the source table.	505

Related reference:

- "Base aggregate table" on page 501
- "Change aggregate table" on page 501
- "Consistent-change data (CCD) table" on page 502
- "Point-in-time table" on page 504
- "Replica table" on page 505
- "User copy table" on page 505

Tables at the Capture control server and their column descriptions

This section provides greater detail of each table stored at the Capture control server. It also lists and briefly describes the columns in each table. The control tables are listed in alphabetical order, and the columns are listed in the order that they appear in each table from left to right.

ASN.IBMSNAP_CAPSCHEMAS

Server: Capture control server

Index: CAP_SCHEMA_NAME

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results while using the administration tools.

The Capture schemas table holds the names of all Capture schemas. It allows the Replication Center and other utilities to quickly find all of the tables for a given Capture control server. A row is automatically inserted each time you create a new Capture schema.

The following two tables show operating system-specific layouts of the Capture schemas table.

Table 67. Columns in the Capture schemas table for all operating systems other than OS/400

Column name	Description
CAP_SCHEMA_NAME	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode; **Nullable**: Yes. The name of a Capture schema. A row exists for each Capture schema.

Table 68. Columns in the Capture schemas table for OS/400

Column name	Description
CAP_SCHEMA_NAME	**Data type**: VARCHAR(30); **Nullable**: Yes. The name of a Capture schema. A row exists for each Capture schema.
STATUS	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the Capture program that is identified by this Capture schema is running: **Y** The Capture program is running. **N** The Capture program is not running.

schema.IBMSNAP_AUTHTKN (OS/400)

Server: Capture control server

Default schema: ASN

Index: JRN_LIB, JRN_NAME

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The Apply-qualifier cross-reference table is used in the OS/400 environment only. This table is used during update-anywhere replication to keep track of the transactions that have been processed by a particular Apply program, which is identified by an Apply qualifier. The Capture program prunes this table based on the retention limit that you set.

Table 69 provides a brief description of the columns in the Apply-qualifier cross-reference table.

Table 69. Columns in the apply-qualifier cross-reference table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. The Apply qualifier that identifies which Apply program processed the transaction. This qualifier is used during update-anywhere replication to prevent the Apply program from replicating the same changes repeatedly.
IBMSNAP_AUTHTKN	**Data type**: CHAR(26); **Nullable**: No. The job name that is associated with the transaction. Capture for iSeries matches the name in this column with the name of the job that issued the transaction to determine whether the transaction was issued by either the Apply program or a user application. If the job names match, then Capture for iSeries copies the Apply qualifier that's in the APPLY_QUAL column of this table to the APPLY_QUAL column in the corresponding row of the UOW table. If the names do not match, then Capture for iSeries sets the APPLY_QUAL column of the UOW row to null. This column is not automatically copied to other tables; you must select it and copy it as a user data column.
JRN_LIB	**Data type**: CHAR(10); **Nullable**: No. The library name of the journal from which the transactions came.
JRN_NAME	**Data type**: CHAR(10); **Nullable**: No. The name of the journal from which the transactions came.
IBMSNAP_LOGMARKER	**Data type**: TIMESTAMP; **Nullable**: No. The approximate time that the transaction was committed at the Capture control server.

schema.IBMSNAP_CAPENQ (UNIX, Windows, z/OS)

Server: Capture control server

Default schema: ASN

Index: None

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The Capture enqueue table is not used on non-DB2 relational or OS/400 servers.

For a single Capture schema, the Capture enqueue table ensures that:

- For DB2 for Linux, UNIX and Windows, only one Capture program is running per database
- For non-data-sharing DB2 for z/OS, only one Capture program is running per subsystem
- For data-sharing DB2 for z/OS, only one Capture program is running per data-sharing group

While running, the Capture program exclusively locks this table.

Table 70 provides a brief description of the column in the Capture enqueue table.

Table 70. Column in the Capture enqueue table

Column name	Description
LOCKNAME	**Data type**: CHAR(9); **Nullable**: Yes. This column contains no data.

schema.IBMSNAP_CAPMON

Server: Capture control server

Default schema: ASN

Index: MONITOR_TIME

The Capture program inserts a row in the Capture monitor table after each interval to provide you with operational statistics. The Replication Center uses information in this table (and in other tables) so you can monitor the status of the Capture program. In the Capture parameters (IBMSNAP_CAPPARMS) table, the value that you specify for MONITOR_INTERVAL indicates how frequently the Capture program makes inserts into the Capture monitor table, and the value that you specify for the MONITOR_LIMIT indicates the number of minutes that rows remain in the table before they are eligible for pruning.

Table 71 provides a brief description of the columns in the Capture monitor table.

Table 71. Columns in the Capture monitor table

Column name	Description
MONITOR_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The timestamp (at the Capture control server) when the row was inserted into this table.
RESTART_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The timestamp when the current invocation of the Capture program was restarted.
CURRENT_MEMORY	**Data type**: INT; **Nullable**: No. The amount of memory (in bytes) that the Capture program used.
CD_ROWS_INSERTED	**Data type**: INT; **Nullable**: No. The number of rows that the Capture program inserted into the CD table for all source tables.
RECAP_ROWS_SKIPPED	**Data type**: INT; **Nullable**: No. For update-anywhere replication, this is the number of rows that the Capture program processed but did not insert into the CD table. The rows were skipped because the registration was defined for the Capture program to not recapture changes that have been replicated to this table that did not originate at this source server.
TRIGR_ROWS_SKIPPED	**Data type**: INT; **Nullable**: No. The number of rows that the Capture program processed but did not insert into the CD table. The rows were skipped because you defined a trigger on the registration for the Capture program to suppress certain rows.

Table 71. Columns in the Capture monitor table (continued)

Column name	Description
CHG_ROWS_SKIPPED	**Data type**: INT; **Nullable**: No. The number of rows that the Capture program processed but did not insert into the CD table. The rows were skipped because the registration was defined for the Capture program to only capture changes that occur in registered columns.
TRANS_PROCESSED	**Data type**: INT; **Nullable**: No. The number of transactions at the source system that the Capture program processed.
TRANS_SPILLED	**Data type**: INT; **Nullable**: No. The number of transactions at the source system that the Capture program spilled to disk due to memory restrictions.
MAX_TRAN_SIZE	**Data type**: INT; **Nullable**: No. The largest transaction that occurred at the source system. Knowing the transaction size might influence you to change the memory parameters.
LOCKING_RETRIES	**Data type**: INT; **Nullable**: No. The number of times a deadlock caused rework.
JRN_LIB (OS/400)	**Data type**: CHAR(10); **Nullable**: Yes. The library name of the journal that the Capture program was processing.
JRN_NAME (OS/400)	**Data type**: CHAR(10); **Nullable**: Yes. The name of the journal that the Capture program was processing.
LOGREADLIMIT	**Data type**: INT; **Nullable**: No. The number of times that the Capture program paused from reading log records because 1000 records had been read, but no completed transactions had yet been encountered within those 1000 records.
CAPTURE_IDLE	**Data type**: INT; **Nullable**: No. The number of times that the Capture program slept because it didn't have any work to process.
SYNCHTIME	**Data type**: TIMESTAMP; **Nullable**: No. The current value of SYNCHTIME read from the global row of the register table when the monitor record was inserted into this table.

schema.IBMSNAP_CAPPARMS

Server: Capture control server

Default schema: ASN

Index: None

This table contains information that you can update by using SQL.

The Capture parameters table contains parameters that you can modify to control the operations of the Capture program. You can define these parameters to set values such as the length of time that the Capture program retains data in the CD and UOW tables before pruning and the amount of time that the Capture program

is allowed to lag in processing log records. If you make changes to the parameters in this table, the Capture program reads your modifications only during startup.

Table 72 provides a brief description of the columns in the Capture parameters table.

Table 72. Columns in the Capture parameters table

Column name	Description
RETENTION_LIMIT	**Data type**: INT; **Nullable**: Yes. The length of time that rows remain in the CD, UOW, and signal tables before they become eligible for pruning, in cases where they have not been pruned based on the normal criteria. Normally, CD and UOW rows are pruned after they are applied to all targets, and signal rows are pruned when their cycle is complete (SIGNAL_STATE = C).
LAG_LIMIT	**Data type**: INT; **Nullable**: Yes. The number of minutes that the Capture program is allowed to lag when processing log records before it shuts itself down. During periods of high update frequency, full refreshes can be more economical than updates.
COMMIT_INTERVAL	**Data type**: INT; **Nullable**: Yes. How often, in seconds, the Capture program commits data to the Capture control tables, including the UOW and CD tables. This value should be less than the DB2 lockout value to prevent contention between the Capture and pruning threads.
PRUNE_INTERVAL	**Data type**: INT; **Nullable**: Yes. How often, in seconds, the Capture program automatically prunes (AUTOPRUNE = Y) rows in the CD, UOW, signal, trace, and Capture monitor tables that are no longer needed. A lower prune interval saves space, but increases processing costs. A higher prune interval requires more CD and UOW table space, but decreases processing costs.
TRACE_LIMIT	**Data type**: INT; **Nullable**: Yes. The number of minutes that rows remain in the Capture trace (IBMSNAP_CAPTRACE) table before they are eligible for pruning. During the pruning process, the rows in the Capture trace table are pruned if the number of minutes (current timestamp – the time a row was inserted in the Capture trace table) exceeds the value of TRACE_LIMIT.
MONITOR_LIMIT	**Data type**: INT; **Nullable**: Yes. The number of minutes that rows remain in the Capture monitor (IBMSNAP_CAPMON) table before they are eligible for pruning. During the pruning process, rows in the Capture monitor table are pruned if the value of minutes (current timestamp – MONITOR_TIME) exceeds the value of MONITOR_LIMIT.
MONITOR_INTERVAL	**Data type**: INT; **Nullable**: Yes. How often, in seconds, that the monitor thread adds a row to the Capture monitor (IBMSNAP_CAPMON) table. For Capture for iSeries, enter an interval greater than 120 seconds.
MEMORY_LIMIT	**Data type**: SMALLINT; **Nullable**: Yes. The amount of memory, in megabytes, that the Capture program is allowed to use. After this allocation is used up, memory transactions will spill to a file.

Table 72. Columns in the Capture parameters table (continued)

Column name	Description
REMOTE_SRC_SERVER	**Data type**: CHAR(18); **Nullable**: Yes. Reserved for future options of DB2 replication. Currently this column contains the default value of null.
AUTOPRUNE	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the Capture program automatically prunes rows that are no longer needed from the CD, UOW, signal, trace, and Capture monitor tables: **Y** Autopruning is on. **N** Autopruning is off.
TERM	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the Capture program terminates when DB2 terminates: **Y** The Capture program terminates when DB2 terminates. **N** The Capture program stays active and waits for DB2 to be restarted.
AUTOSTOP	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the Capture program stops capturing changes as soon as it reaches the end of the active logs: **Y** The Capture program stops as soon as it reaches the end of the active logs. **N** The Capture program continues running when it reaches the end of the active logs.
LOGREUSE	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the Capture program overwrites the Capture log file or appends to it. **Y** The Capture program reuses the log file by first deleting it and then recreating it when the Capture program is restarted. **N** The Capture program appends new information to the Capture log file.
LOGSTDOUT	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates where the Capture program directs the log file messages: **Y** The Capture program directs log file messages to both the standard out (STDOUT) and the log file. **N** The Capture program directs most log file messages to the log file only. Initialization messages go to both the standard out (STDOUT) and the log file.
SLEEP_INTERVAL (Linux, UNIX, Windows, z/OS)	**Data type**: SMALLINT; **Nullable**: Yes. The number of seconds that the Capture program sleeps when it reaches the end of the active logs (in Linux, UNIX and Windows, or in z/OS non-data-sharing environments), or when an inefficient amount of data has been returned (in z/OS data-sharing environments).
CAPTURE_PATH	**Data type**: VARCHAR(1040); **Nullable**: Yes. The path where the output from the Capture program is sent.

Table 72. Columns in the Capture parameters table (continued)

Column name	Description
STARTMODE	**Data type**: VARCHAR(10); **Nullable**: Yes. The processing procedure that the Capture program uses when it is started: **cold** The Capture program deletes all rows in its CD tables and UOW table during initialization. All subscriptions to these replication sources are fully refreshed during the next Apply processing cycle (that is, all data is copied from the source tables to the target tables). If the Capture program tries to cold start but you disabled full refresh, the Capture program will start but the Apply program will fail and will issue an error message. **warmsi** The Capture program warm starts; except if this is the first time you are starting the Capture program then it switches to a cold start. The warmsi start mode ensures that cold starts happen only when you initially start the Capture program. **warmns** The Capture program warm starts. If it can't warm start, it does *not* switch to cold start. The warmns start mode prevents cold starts from occurring unexpectedly and is useful when problems arise (such as unavailable databases or table spaces) that require repair and that prevent a warm start from proceeding. When the Capture program warm starts, it resumes processing where it ended. If errors occur after the Capture program started, the Capture program terminates and leaves all tables intact. **warmsa** If warm start information is available, the Capture program resumes processing where it ended in its previous run. If the Capture program cannot warm start, it switches to a cold start that refreshes all of your target tables.

schema.IBMSNAP_CAPTRACE (DB2 only)

Server: Capture control server

Default schema: ASN

Index: TRACE_TIME

The Capture trace table contains important messages from the Capture program.

The following two tables show operating system-specific layouts of the Capture trace table.

Table 73. Columns in the Capture trace table for Linux, UNIX, Windows, and z/OS

Column name	Description
OPERATION	**Data type**: CHAR(8); **Nullable**: No. The type of Capture program operation, for example, initialization, capture, or error condition.
TRACE_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The time at the Capture control server that the row was inserted in the Capture trace table.

Table 73. Columns in the Capture trace table for Linux, UNIX, Windows, and z/OS (continued)

Column name	Description
DESCRIPTION	**Data type**: VARCHAR(1024); **Nullable**: No. The message ID followed by the message text. It can be an error message, a warning message, or an informational message. This column contains English-only text.

Table 74. Columns in the Capture trace table for OS/400

Column name	Description
OPERATION	**Data type**: CHAR(8); **Nullable**: No. The type of operation that the Capture program performed, for example, initialization, capture, or error condition.
TRACE_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The time that the row was inserted in the Capture trace table. TRACE_TIME rows that are eligible for trace limit pruning will be deleted when the Capture program prunes the CD and UOW tables.
JOB_NAME	**Data type**: CHAR(26); **Nullable**: No. The fully qualified name of the job that wrote this trace entry. **Position** — **Description** **1–10** The Capture schema name or the journal job name **11–20** The ID of the user who started the Capture program **21–26** The job number
JOB_STR_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The starting time of the job that is named in the JOB_NAME column.
DESCRIPTION	**Data type**: VARCHAR(298); **Nullable**: No. The message ID followed by the message text. The message ID is the first seven characters of the DESCRIPTION column. The message text starts at the ninth position of the DESCRIPTION column.

schema.CCD_table (non-DB2)

Server: Capture control server

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause a loss of data.

Consistent-change-data (CCD) tables at the Capture control server are tables that contain information about changes that occur at a non-DB2 source and additional columns to identify the sequential ordering of those changes. A CCD table at the Capture control server is a table that is populated by a program other than the Apply program. It can be either:

- An internal CCD table for a non-DB2 relational source.

  For change-capture replication, the Capture triggers insert changes in this table as updates occur at the non-DB2 relational source. The name of this type of CCD table is stored on the same row in the register (IBMSNAP_REGISTER) table as

the replication source that it holds changes from. This table is automatically pruned by the pruning trigger that is created when you register a non-DB2 relational source.

- An external CCD table for non-relational and multi-vendor data.

  External programs can create CCD tables to be used by DB2 replication as replication sources. These external programs capture IMS changes in a CCD table, so that copies of IMS data can be recreated in a relational database. The external programs must initialize, maintain, and supply the correct values for the control columns. If you have externally populated CCD tables that are not maintained by a program such as IMS DataPropagator or DataRefresher, you must maintain these tables yourself so that the Apply program can read the CCD tables as sources and function correctly. For details on how to maintain an externally populated CCD, see "Maintaining CCD tables as sources (IMS)" on page 55.

For information on CCD tables as targets in a subscription-set member, see "Consistent-change data (CCD) table" on page 502.

Table 75 provides a brief description of the columns in the CCD table.

Table 75. Columns in the CCD table

Column name	Description
IBMSNAP_INTENTSEQ	A sequence number that uniquely identifies a change. This value is globally ascending.
IBMSNAP_OPERATION	A flag that indicates the type of operation for a record: **I** Insert **U** Update **D** Delete
IBMSNAP_COMMITSEQ	A sequence number that provides transactional order.
IBMSNAP_LOGMARKER	The time that the data was committed.
user key columns	If the CCD table is condensed, this column contains the columns that make up the target key.
user non-key columns	The non-key data columns from the source table. The column names that are in the source table do not need to match these column names, but the data types must be compatible.
user computed columns	User-defined columns that are derived from SQL expressions. You can use computed columns with SQL functions to convert source data types to different target data types.

Related reference:

- "Consistent-change data (CCD) table" on page 502

schema.CD_table

Server: Capture control server

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause a loss of data.

Change-data (CD) tables record all committed changes made to a replication source. Pruning of the CD table is coordinated by the prune set (IBMSNAP_PRUNE_SET) table. (See "*schema*.IBMSNAP_PRUNE_SET" on page 447

for more information about how the CD table gets pruned.) Unlike other Capture control tables, CD tables are created when you define a replication source; they are not created automatically when you generate the control tables for the Capture control server.

Table 76 provides a list and a brief description of the columns in the CD table.

Table 76. Columns in the CD table

Column name	Description
IBMSNAP_COMMITSEQ	The log sequence number of the captured commit statement. This column, which is also in the UOW table, is included in the CD table to allow the Apply program to process user copy target tables without needing to join the CD table with the UOW table. In cases where a join between the CD table and the UOW table is required, the join is done using the IBMSNAP_COMMITSEQ column.
IBMSNAP_INTENTSEQ	The log sequence number of the log record of the change (insert, update, or delete). This value is globally ascending. If you selected for updates to be processes as delete/insert pairs, the IBMSNAP_INTENTSEQ value for the delete row is manufactured to be slightly smaller than the corresponding value for the insert row.
IBMSNAP_OPERATION	A flag that indicates the type of operation for a record: **I** Insert **U** Update **D** Delete
user column after-image	In most cases, the after-image column contains the value that is in the source column after the change occurs. This column has the same name, data type, and null attributes as the source column. In the case of an update, this column reflects the new value of the data that was updated. In the case of a delete, this column reflects the value of the data that was deleted. In the case of an insert, this column reflects the value of the data that was inserted.
user column before-image	This column only exists in the CD table if you registered the source to include before-image column values. In most cases, the before-image column contains the value that was in the source column before the change occurred. This column has the same name as the source column, prefixed by the value in the BEFORE_IMG_PREFIX column in the register (IBMSNAP_REGISTER) table. It also has the same data type as the source column; however, it always allows null values for insert operations regardless of the source column's null attributes. In the case of an update, this column reflects the data that was updated. In the case of a delete, this column reflects the data that was deleted. In the case of an insert, this column is null.

schema.IBMSNAP_PARTITIONINFO

Server: Capture control server

Default schema: ASN

Index: PARTITIONID, USAGE

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data. If you delete the row from this table, the Capture program is forced to cold start.

The partition information table augments the restart (IBMSNAP_RESTART) table in a multi-partitioned environment, and contains information that enables the Capture program to restart from the earliest required log sequence number within each

partition's set of log files. In a multi-partitioned environment, the partitioninfo table and the restart table replace the warm_start table from DB2 replication Version 7 and earlier versions. A row is inserted into this table every time a partition is added. The Capture program will start reading the log file of any new partitions from the first log sequence number that DB2 used after the first database CONNECT was issued.

If you have never started the Capture program, then this table is empty, and the Capture program must perform a cold start.

Table 77 provides a brief description of the columns in the partition information table.

Table 77. Columns in the partitioninfo table

Column name	Description
PARTITIONID	**Data type**: INT; **Nullable**: No. The partition ID for each valid partition.
USAGE	**Data type**: CHAR(1); **Nullable**: No. The usage of the log sequence number (LSN). An ″R″ in this column indicates that the LSN has been restarted.
SEQUENCE	**Data type**: CHAR(10) for bit data; **Nullable**: No. The restart LSN for the node that has the partition ID.
STATUS	**Data type**: CHAR(1); **Nullable**: Yes. The status of the partition. An ″A″ in this column indicates that the partition is active. This column is reserved for future use.
LAST_UPDATE	**Data type**: TIMESTAMP; **Nullable**: Yes. The timestamp when the restart LSN for the node that has the partition ID was last updated.

schema.IBMSNAP_PRUNCNTL

Server: Capture control server

Default schema: ASN

Index: SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL, APPLY_QUAL, SET_NAME, TARGET_SERVER, TARGET_TABLE, TARGET_OWNER

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The pruning control table contains detailed information regarding all subscription set members that are defined for this Capture schema. This table is used in conjunction with the prune set (IBMSNAP_PRUNE_SET) table during pruning. It is also used during the initialization handshake process between the Apply and Capture programs.

For DB2 sources, you can invoke pruning by issuing the **prune** command or have it done automatically. See "*schema*.IBMSNAP_CAPPARMS" on page 437 for more

information about using the Capture parameters table to set AUTOPRUNE.For non-DB2 relational sources, pruning is done by the pruning trigger that was created when you registered the source.

Table 78 provides a brief description of the columns in the pruning control table.

Table 78. Columns in the pruning control table

Column name	Description
TARGET_SERVER	**Data type**: CHAR(18); **Nullable**: No. The server name where target table or view for this member resides.
TARGET_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode; **Nullable**: No. The high-level qualifier for the target table or view for this member.
TARGET_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No. The name of the target table or view for this member.
SYNCHTIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The Capture program sets this timestamp during the initialization handshake process with the Apply program. The value comes from the timestamp of the commit log record that is associated with the transaction of the CAPSTART signal insert. It will not be updated again unless a subsequent initialization process takes place.
SYNCHPOINT	**Data type**: CHAR(10) for bit data; **Nullable**: Yes. The Capture program sets this value during the initialization handshake process with the Apply program. The value comes from the log sequence number of the commit log record that is associated with the transaction of the CAPSTART signal insert. It will not be updated again unless a subsequent initialization process takes place.
SOURCE_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode; **Nullable**: No. The high-level qualifier of the source table or view for this member.
SOURCE_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No. The name of the source table or view for this member.
SOURCE_VIEW_QUAL	**Data type**: SMALLINT; **Nullable**: No. This column is used to support multiple registrations for different source views with identical SOURCE_OWNER and SOURCE_TABLE column values. This value is set to 0 for physical tables that are defined as sources and is greater than 0 for views that are defined as sources.
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. The Apply qualifier that identifies which Apply program is processing this member.
SET_NAME	**Data type**: CHAR(18); **Nullable**: No. The name of the subscription set that this subscription-set member belongs to.

Table 78. Columns in the pruning control table (continued)

Column name	Description
CNTL_SERVER	**Data type**: CHAR(18); **Nullable**: No. The name of the server where the Apply control tables reside for this Apply program, which is identified by the APPLY_QUAL.
TARGET_STRUCTURE	**Data type**: SMALLINT; **Nullable**: No. A value that identifies the type of target table or view: **1** Source table **3** CCD table **4** Point-in-time table **5** Base aggregate table **6** Change aggregate table **7** Replica table **8** User copy table
CNTL_ALIAS	**Data type**: CHAR(8); **Nullable**: Yes. The DB2 Universal Database alias corresponding to the Apply control server named in the CNTL_SERVER column.
PHYS_CHANGE_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode; **Nullable**: Yes. The value in the PHYS_CHANGE_OWNER column from the register (IBMSNAP_REGISTER) table that is associated with the source of this particular subscription-set member.
PHYS_CHANGE_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier;**Nullable**: Yes. The value in the PHYS_CHANGE_TABLE column from the register (IBMSNAP_REGISTER) table that is associated with the source of this particular subscription-set member.
MAP_ID	**Data type**: VARCHAR(10); **Nullable**: No. A uniqueness factor that provides a shorter, more easily used index into this table. It is also used to associate CAPSTART inserts into the signal table with the appropriate row in the pruning control table.

schema.IBMSNAP_PRUNE_LOCK

Server: Capture control server

Default schema: ASN

Index: None

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The prune lock table is used to serialize the access of CD tables during a cold start or retention-limit pruning. This table ensures that the Apply program does not access the CD table during these critical phases. There are no rows in this table.

schema.IBMSNAP_PRUNE_SET

Server: Capture control server

Default schema: ASN

Index: TARGET_SERVER, APPLY_QUAL, SET_NAME

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The prune set table tracks the progress of the Capture and Apply programs for each subscription set to help coordinate the pruning of the CD and UOW tables. Unlike the pruning control (IBMSNAP_PRUNCNTL) table, which has one row for each source-to-target mapping, the prune set table has one row for each subscription set.

Table 79 provides a brief description of the columns in the prune set table.

Table 79. Columns in the prune set table

Column name	Description
TARGET_SERVER	**Data type**: CHAR(18); **Nullable**: No. The server name where target tables or views for this set reside.
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. The Apply qualifier that identifies which Apply program is processing this set.
SET_NAME	**Data type**: CHAR(18); **Nullable**: No. The name of the subscription set.
SYNCHTIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The Apply program uses this column to record its progress, indicating that it has processed data up to this timestamp for the subscription set.
SYNCHPOINT	**Data type**: CHAR(10) for bit data; **Nullable**: No. The Apply program uses this column to record its progress, indicating that it has processed data up to this synchpoint value for the subscription set.

schema.IBMSNAP_REG_EXT (OS/400)

Server: Capture control server

Default schema: ASN

Index: VERSION, SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The register extension table is an OS/400-specific table that provides supplemental information for the register (IBMSNAP_REGISTER) table. For every row in the register table, there is a matching row in the register extension table containing a few additional OS/400-specific columns.

IBMSNAP_REG_EXT

This table is maintained by a trigger program (program QZSNJLV8 in library QDP4) on the register table. The trigger is defined at the time the register table is created.

The information from this table is used to track where and how you defined your replication sources on an OS/400 server.

Table 80 provides a brief description of the columns in the register extension table.

Table 80. Columns in the register extension table

Column name	Description
VERSION	**Data type**: INT; **Nullable**: No. The version of DB2 DataPropagator for iSeries that you used to register the source.
SOURCE_OWNER	**Data type**: VARCHAR(30); **Nullable**: No. The high-level qualifier of the source table or view that you registered.
SOURCE_TABLE	**Data type**: VARCHAR(128); **Nullable**: No. The name of the source table or view that you registered.
SOURCE_NAME	**Data type**: CHAR(10); **Nullable**: Yes. A ten-character system name of the source table or view that you used to issue the commands.
SOURCE_MBR	**Data type**: CHAR(10); **Nullable**: Yes. The name of the source table member, which is used for issuing Receive Journal Entry (**RCVJRNE**) commands and ALIAS support.
SOURCE_TABLE_RDB	**Data type**: CHAR(18); **Nullable**: Yes. When using remote journals, this column contains the database name of the system where the source table actually resides. For local journals, this column is null.
JRN_LIB	**Data type**: CHAR(10); **Nullable**: Yes. The library name of the journal that the source table uses.
JRN_NAME	**Data type**: CHAR(10); **Nullable**: Yes. The name of the journal that is used by a source table. An asterisk followed by nine blanks in this column means that the source table is currently not in a journal, and it is not possible for the Capture program to capture data for this source.
FR_START_TIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The time when the Apply program began to perform a full refresh.
SOURCE_VIEW_QUAL	**Data type**: SMALLINT; **Nullable**: No. Supports the view of subscriptions by matching the similar column in the register table. This value is set to equal 0 for physical tables that are defined as a source and is greater than 0 for views that are defined as sources. You must have this column to support multiple subscriptions for different source views containing identical SOURCE_OWNER and SOURCE_TABLE column values.

Table 80. Columns in the register extension table (continued)

Column name	Description
CMT_BEHAVIOR_CASE	**Data type**: SMALLINT; **Nullable**: No, with default; **Default**: 0. An integer that represents how the application programs that are updating the source table use commitment control. The Capture program uses this value to manage its memory usage for CD rows that it has constructed but is not yet ready to write to the CD tables. **-1** The applications' commitment control pattern is not yet established. This is the initial value in the column. **0** None of the applications that update the source uses commitment control. **1** All of the applications that update the source use commitment control. Therefore, two different applications never update the same source table under commitment control at the same time. **2** For concurrent applications that update the source, some use commitment control and others do not. It is possible that there are two applications updating the source table using commitment control concurrently.
MAX_ROWS_BTWN_CMTS	**Data type**: SMALLINT; **Nullable**: No, with default; **Default**: 0. The maximum number of rows that the Capture program can process before it commits data to the CD table.

schema.IBMSNAP_REGISTER

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

Server: Capture control server

Default schema: ASN

Index: SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL

The register table contains information about replication sources, such as the names of the replication source tables, their attributes, and the names of the CD and CCD tables associated with them. A row is automatically inserted into this table every time you define a new replication source table or view for the Capture program to process.

The register table is the place you should look if you need to know how you defined your replication sources.

Table 81 provides a brief description of the columns in the register table.

Table 81. Columns in the register table

Column name	Description
SOURCE_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: No. The high-level qualifier of the source table or view that you registered.

Table 81. Columns in the register table (continued)

Column name	Description
SOURCE_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No. The name of the source table or view that you registered.
SOURCE_VIEW_QUAL	**Data type**: SMALLINT; **Nullable**: No. This column is used to support multiple registrations for different source views with identical SOURCE_OWNER and SOURCE_TABLE column values. This value is set to 0 for physical tables that are defined as sources, and is greater than 0 for views that are defined as sources.
GLOBAL_RECORD	**Data type**: CHAR(1); **Nullable**: No.
SOURCE_STRUCTURE	**Data type**: SMALLINT; **Nullable**: No. A value that identifies the structure of the source table or view: **1** User table **3** CCD table **4** Point-in-time table **5** Base aggregate table **6** Change aggregate table **7** Replica table **8** User copy table
SOURCE_CONDENSED	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates whether the source table is a condensed table, meaning that all rows with the same key are condensed to one row: **Y** The source is condensed. **N** The source is not condensed. **A** The source is a base-aggregate or change-aggregate table.
SOURCE_COMPLETE	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates how the source table stores rows of primary key values: **Y** The source table contains a row for every primary key value of interest. **N** The source table contains a subset of rows of primary key values.
CD_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: Yes. The high-level qualifier of the source's CD table. **For tables as sources** For all registered source tables that are not external CCD tables, this column contains the high-level qualifier of the CD table associated with that source table. **For views as sources** This column contains the high-level qualifier of the CD view. **For external CCD tables as sources** This column is null.

Table 81. Columns in the register table (continued)

Column name	Description
CD_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: Yes. The name of the source's CD table. **For tables as sources** For all registered source tables that are not external CCD tables, this column contains the name of the CD table that holds captured updates of the source table. **For views as sources** This column contains the name of the CD view. **For external CCD tables as sources** This column is null.
PHYS_CHANGE_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: Yes. The high-level qualifier of the table or view that the Apply program uses for change-capture replication: **For tables as sources** For all registered source tables that are not external CCD tables, this column contains the high-level qualifier of the physical CD table that is associated with that source table. **For views as sources** This column contains the high-level qualifier of the physical CD table that is associated with that source view. **For external CCD tables as sources** This column contains the high-level qualifier of the external CCD table.
PHYS_CHANGE_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: Yes. The name of the table or view that the Apply program uses for change-capture replication: **For tables as sources** For all registered source tables that are not external CCD tables, this column contains the name of the physical CD table that is associated with that source table. **For views as sources** This column contains the name of the physical CD table that is associated with that source view. **For external CCDs as sources** This column contains the name of the external CCD table.
CD_OLD_SYNCHPOINT	**Data type**: CHAR(10) for bit data; **Nullable**: Yes. This column is used for the initial handshake between the Apply program and the Capture program. The Capture program then begins capturing data from this log sequence number in the source log. This column is also used to show that retention-limit pruning has occurred for a CD table. If this value is null, then the registration is inactive.
CD_NEW_SYNCHPOINT	**Data type**: CHAR(10) for bit data; **Nullable**: Yes. The Capture program advances this column as it inserts new rows into the CD table. The Apply program uses this column to see if there are new changes to be replicated.

Table 81. Columns in the register table (continued)

Column name	Description
DISABLE_REFRESH	**Data type**: SMALLINT; **Nullable**: Yes. A flag that indicates whether full refreshes are allowed: **0** Full refreshes are allowed. **1** Full refreshes are prevented.
CCD_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: Yes. For a source that has an internal CCD table associated with it, this column contains the high-level qualifier of the internal CCD. For an external CCD table, this column is null.
CCD_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: Yes. For a source that has an internal CCD table associated with it, this column contains the name of the internal CCD. For an external CCD table, this column is null.
CCD_OLD_SYNCHPOINT	**Data type**: CHAR(10) for bit data; **Nullable**: Yes. The log sequence number when the CCD table was reinitialized. This column is related to full-refresh processing against CCD tables. The value in this column needs to be changed only when the CCD table is initially or subsequently fully refreshed. This value can be much older than any row remaining in the CCD table. If this column is not maintained, the Apply program using the CCD table as a replication source will not know that the CCD table was reinitialized, so it will fail to reinitialize complete copies of the CCD source.
SYNCHPOINT	**Data type**: CHAR(10) for bit data; **Nullable**: Yes. In the global row (where GLOBAL_RECORD = Y), the synchpoint represents the log sequence number of the last log or journal record processed by the Capture program. In any row in the IBMSNAP_REGISTER table that contains registration information about a CCD table (internal or external), the synchpoint value is advanced by the program that maintains the CCD table to indicate that there is new data available in that CCD table.
SYNCHTIME	**Data type**: TIMESTAMP; **Nullable**: Yes. In the global row (where GLOBAL_RECORD = Y), the synchtime represents the timestamp from the last log or journal record processed by the Capture program. If the Capture program has reached the end of the DB2 log, the synchtime is advanced to the current DB2 timestamp. In any row in the IBMSNAP_REGISTER table that contains registration information about a CCD table (internal or external), the synchtime value is advanced by the program that maintains the CCD table to indicate the currency of data available in that CCD table.
CCD_CONDENSED	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the internal CCD that is associated with this source is condensed, meaning that all rows with the same key are condensed to one row: **Y** The internal CCD is condensed. **N** The internal CCD is not condensed. **NULL** No internal CCD table is defined for this source.

Table 81. Columns in the register table (continued)

Column name	Description
CCD_COMPLETE	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the internal CCD table that is associated with this source is complete, meaning that it initially contained all the rows from the source table: **N** The internal CCD is not complete. **NULL** No internal CCD table is defined for this source.
ARCH_LEVEL	**Data type**: CHAR(4); **Nullable**: No. The architectural level of the replication control tables: **0801** Version 8 SQL Replication **0803** Version 8 SQL Replication with enhanced support for Oracle sources **0805** Version 8 SQL Replication with support for DB2 for z/OS new-function mode
DESCRIPTION	**Data type**: CHAR(254); **Nullable**: Yes. A description of the replication source.
BEFORE_IMG_PREFIX	**Data type**: VARCHAR(4); **Nullable**: Yes. The one-character prefix that identifies before-image column names in the CD table. The combination of the before-image prefix and the CD column name must be unambiguous, meaning that a prefixed CD column name cannot be the same as a current or potential after-image column name. The length in bytes of the BEFORE_IMG_PREFIX is: **1** For an ASCII or an EBCDIC single byte prefix character. **2** For an ASCII double byte prefix character. **4** For an EBCDIC DBCS prefix character. This length allows for shift-in and shift-out characters.
CONFLICT_LEVEL	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates the level of conflict detection for this source: **0** The Apply program does not check for conflicts. Data consistency must be enforced by your application to avoid potential conflicting updates. **1** Standard detection with cascading transaction rejection. The Apply program checks for conflicts based on the changes captured to this point. The Apply program will reverse any conflicting transaction at the replica, as well as any transactions with dependencies on the conflicting transaction. Changes captured after the Apply program begins conflict detection will not be checked during this Apply cycle. **2** Enhanced detection with cascading transaction rejection. The Apply program waits until the Capture program captures all changes from the log or journal (see description of the SYNCHTIME column) and then does a standard conflict detection as when set to 1. During the wait time, the Apply program holds a LOCK on the source tables to ensure that no changes are made during the conflict detection process.

Table 81. Columns in the register table (continued)

Column name	Description
CHG_UPD_TO_DEL_INS	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates how the Capture program stores updates in the CD table. **Y** The Capture program stores updates using two rows in the CD table, one for the delete and one for the insert. The Apply program processes the delete first and the insert second. When this Y flag is set, every update to a replication source is stored in the CD table using two rows. This flag ensures that updates made to partitioning columns or columns referenced by a subscription-set predicate are processed correctly. **N** Each update to the source table is stored in a single row in the CD table.
CHGONLY	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the Capture program captures all changes that occur at the source or only changes that occur in registered columns. Typically you should have this option set to Y to minimize the number of rows that the Capture program inserts into the CD table, but you might want to set this option to N in order to track exactly which rows in the source table were updated. For example, you might just be capturing the primary key column values to audit which rows have been changed in a source table. **Y** The Capture program only captures changes that occur in registered columns in the source table. **N** The Capture program captures changes from all columns in the source table.
RECAPTURE	**Data type**: CHAR(1); **Nullable**: Yes. This column is for update-anywhere replication and contains a flag that indicates whether changes that originate from a table or view are recaptured and forwarded to other tables or views. For tables at the master site: **N** Updates to the master that were applied from a replica are not recaptured and will not be replicated to other replicas. **Y** Updates to the master that were applied from a replica and will be replicated to other replicas. For tables at a replica site: **Y** Updates to the replica that were applied from the master are recaptured and are available to be replicated to another table that uses the replica as its source. **N** Updates to the replica that were applied from the master are not recaptured.
OPTION_FLAGS	**Data type**: CHAR(4); **Nullable**: No. Reserved for future options of DB2 replication. Currently this column contains the default value of NNNN.

Table 81. Columns in the register table (continued)

Column name	Description
STOP_ON_ERROR	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** Y. A flag that indicates whether the Capture program will terminate or just stop processing the registration if it encounters errors while trying to start, initiate, reinitiate, or insert a row into the CD table: **Y** The Capture program terminates when an error occurs while it is trying to start, initiate, reinitiate, or insert a row into the CD table. **N** The Capture program stops the registration but does not terminate when an error occurs while it is trying to start, initiate, reinitiate, or insert a row into the CD table; it continues to process other registrations.
STATE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** I. A flag that indicates what state the registration is in: **S** The Capture program has stopped processing this registration. The Apply program will not work with this registration until you repair the registration and place it in the I (inactive) state. **A** The registration is active. **I** The registration is inactive.
STATE_INFO	**Data type**: CHAR(8); **Nullable**: Yes; If the Capture program stopped processing the registration, this column contains the error message that was issued regarding the failure.

schema.IBMSNAP_REG_SYNCH (non-DB2 relational)

Server: Capture control server

Default schema: ASN

Index: TRIGGER_ME

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The register synchronization table uses an update trigger to initiate an update of the SYNCHPOINT value for all the rows in the register (IBMSNAP_REGISTER) table when the Apply program is preparing to fetch data from a non-DB2 relational data source.

Table 82 provides a brief description of the columns in the register synchronization table.

Table 82. Register synchronization table columns

Column name	Description
TRIGGER_ME	**Data type**: CHAR(1); **Nullable**: No. A flag of Y that indicates whether a trigger was initiated to update the SYNCHPOINT value for all rows in the register table.

Table 82. Register synchronization table columns (continued)

Column name	Description
TIMESTAMP	For Microsoft SQL Server and Sybase sources, this column contains the unique number that is generated by the system when an update occurs on a timestamp column at that table. This value is used to derive the SYNCHPOINT value that is recorded in the register (IBMSNAP_REGISTER) table.

schema.IBMSNAP_RESTART

Server: Capture control server

Default schema: ASN

Index: None

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data. If you delete the row from this table, the Capture program is forced to cold start.

The restart table contains information that enables the Capture program to restart from the earliest required log or journal record. This table replaces the warm start table from DB2 replication Version 7 and earlier versions. It contains one row, which is updated at every commit point; therefore, the Capture program can always restart from exactly the right place without recapturing information that it already processed and inserted into the CD and UOW tables.

If you have never started the Capture program, then this table is empty and the Capture program must perform a cold start.

The following two tables show operating system-specific layouts of the restart table:

Table 83. Columns in the restart table for Linux, UNIX, Windows, and z/OS

Column name	Description
MAX_COMMITSEQ	**Data type**: CHAR(10) for bit data; **Nullable**: No. The maximum logical log sequence number value (IBMSNAP_COMMITSEQ) that the Capture program has committed to the CD and UOW tables.
MAX_COMMIT_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The timestamp that is associated with the log sequence number in the MAX_COMMITSEQ column.
MIN_INFLIGHTSEQ	**Data type**: CHAR(10) for bit data; **Nullable**: No. The logical log sequence number at which the Capture program starts during a warm restart. This value represents the earliest log sequence number that the Capture program found for which a commit or abort record has not yet been found.
CURR_COMMIT_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The local current timestamp when this table was updated by the Capture program.

Table 83. Columns in the restart table for Linux, UNIX, Windows, and z/OS (continued)

Column name	Description
CAPTURE_FIRST_SEQ	**Data type**: CHAR(10) for bit data; **Nullable**: No. The logical log sequence number that is associated with the recovery log that the Capture program started from during the last cold start that the Capture program performed. This value is used to detect if a database RESTORE occurred, which might require the Capture program to perform a cold start because the database log manager might reuse the log sequence numbers during certain RESTORE operations.

For OS/400, the restart table is used to determine the starting time of the **RCVJRNE** (Receive Journal Entry) command. A row is inserted into the restart table for each journal that is used by a replication source or a group of replication sources.

Index: JRN_LIB, JRN_NAME

Table 84. Columns in the restart table for OS/400

Column name	Description
MAX_COMMITSEQ	**Data type**: CHAR(10) for bit data; **Nullable**: No. The journal record number of the most current commit from the UOW table.
MAX_COMMIT_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The timestamp that is associated with the journal record number in the MAX_COMMITSEQ column, or the current timestamp if the Capture program is caught up with the logs and has no work to perform.
MIN_INFLIGHTSEQ	**Data type**: CHAR(10) for bit data; **Nullable**: No. The logical log sequence number that the Capture program starts from during a warm restart.
CURR_COMMIT_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The current timestamp at the point when this table is updated.
CAPTURE_FIRST_SEQ	**Data type**: CHAR(10) for bit data; **Nullable**: No. The journal record number that the Capture program starts from after a cold start.
UID	**Data type**: INTEGER; **Nullable**: No. A unique number that is used as a prefix for the contents of the IBMSNAP_UOWID column located in the UOW table.
SEQNBR	**Data type**: BIGINT; **Nullable**: No. The sequence number of the last journal entry that the Capture program processed.
JRN_LIB	**Data type**: CHAR(10); **Nullable**: No. The library name of the journal that the Capture program is processing.
JRN_NAME	**Data type**: CHAR(10); **Nullable**: No. The name of the journal that the Capture program is processing.

Table 84. Columns in the restart table for OS/400 (continued)

Column name	Description
STATUS	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the Capture program is processing a particular journal job: **Y** The Capture program is processing the journal job. **N** The Capture program is not processing the journal job.

schema.IBMSNAP_SEQTABLE (Informix)

Server: Capture control server

Default schema: ASN

Unique index: SEQ

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The sequencing table contains a sequence of unique numbers that DB2 replication uses as the equivalent of log sequence numbers for Informix tables. These unique identifiers are used in the register (IBMSNAP_REGISTER) table in place of synchpoint values so that the Capture program, Apply program, and Replication Alert Monitor can communicate how far along they have gotten during their last cycle.

Table 85 provides a brief description of the column in the sequencing table.

Table 85. Column in the sequencing table

Column name	Description
SEQ	**Data type**: INTEGER; **Nullable**: No. A unique number used as the log or journal identifiers (synchpoints) for Informix tables.

schema.IBMSNAP_SIGNAL

Server: Capture control server

Default schema: ASN

Index: SIGNAL_TIME

This table contains information that you can update by using SQL.

The signal table stores signals that prompt the Capture program to perform certain actions. The signals are entered by either you or the Apply program.

The signal table is created with the DATA CAPTURE CHANGES attribute, which means that all insert, update, and delete operations performed on this table are visible to the Capture program as log records read from the DB2 recovery log. The Capture program ignores all update and delete log records for the signal table, but it recognizes all validly created and committed log records of signal inserts as

″signals″ that require its attention. The actions that the Capture program performs for a log record from a signal insert depends on what is specified in the signal table for that insert. The values in the signal table provide the instructions to the Capture program regarding the desired action.

Records in this table with a SIGNAL_STATE value of C for complete or records with a timestamp eligible for retention-limit pruning are deleted when the Capture program prunes.

Table 86 provides a brief description of the columns in the signal table.

Table 86. Columns in the signal table

Column name	Description
SIGNAL_TIME	**Data type**: TIMESTAMP; **Nullable**: No, with default; **Default**: current timestamp. A timestamp that is used to uniquely identify the row. The Capture program uses this unique value to find the correct row in the signal table to indicate when it has completed processing the Capture signal. This timestamp column is created as NOT NULL WITH DEFAULT, and therefore a Capture signal can generally be inserted in such a way that DB2 supplies the current timestamp as the SIGNAL_TIME value.
SIGNAL_TYPE	**Data type**: VARCHAR(30); **Nullable**: No. A flag that indicates the type of signal that was posted: **CMD** A signal posted by you, the Apply program, or another application, which is a well known system command or signal. See the SIGNAL_SUBTYPE column for this table for a list of the available signal subtypes. **USER** A signal posted by you or another user. The Capture program updates the value in the SIGNAL_LSN column with the LSN from the log of when the signal was inserted, and it updates the value in the SIGNAL_STATE column to from P (pending) to R (received).

Table 86. Columns in the signal table (continued)

Column name	Description
SIGNAL_SUBTYPE	**Data type**: VARCHAR(30); **Nullable**: yes. The action that the Capture program performs when a signal from a system command (SIGNAL_TYPE = CMD) occurs. **CAPSTART** The Capture program starts capturing changes at the registered source for a particular subscription-set member, which is identified by the MAP_ID (from the IBMSNAP_PRUNCNTL table) in the SIGNAL_INPUT_IN column. For example, the Apply program issues this signal before it performs a full refresh on all target tables in the set to let the Capture program know that the set is ready to begin change-capture replication. The Apply program posts this signal. **STOP** The Capture program stops capturing changes and terminates. This command can only be issued by you, not the Apply program. **CAPSTOP** The Capture program stops capturing changes for a particular registered source, which is identified by *source_owner.source_table* in the SIGNAL_INPUT_IN column. This command can only be issued by you, not the Apply program. **UPDANY** The Apply program (identified by the Apply qualifier in the SIGNAL_INPUT_IN column) lets the Capture program know that it is working with two Capture programs in an update-anywhere configuration. The Apply program posts this signal. When the signal type is USER, the signal subtype is not used or recognized by the Capture program and therefore is not a required field. It can be set to any value that you want.
SIGNAL_INPUT_IN	**Data type**: VARCHAR(500); **Nullable**: yes. If the SIGNAL_TYPE = USER, then this column contains user-defined input. If the SIGNAL_TYPE = CMD, then the meaning of this value depends on the SIGNAL_SUBTYPE for this signal: **CMD + CAPSTART** The mapping identifier. Because the Capture triggers and not the Capture program process non-DB2 relational sources, there is a trigger called SIGNAL_TRIGGER that fires after the IBMSNAP_SIGNAL table is updated, which updates the prune control (IBMSNAP_PRUNCNTL) table with the next value in the sequence. **CMD + UPDANY** The Apply qualifier that identifies the Apply program in the update-anywhere configuration. **CMD + CAPSTOP** The name of the source owner and source table that the Capture program should stop capturing changes for. (*source_owner.source_table*)

Table 86. Columns in the signal table (continued)

Column name	Description
SIGNAL_STATE	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates the status of the signal: **P** The signal is pending; the Capture program has not received it yet. When you post a signal, set the SIGNAL_STATE to P. **R** The Capture program has received the signal. The Capture program sets the SIGNAL_STATE set to R (instead of changing it to C for complete) when it receives a signal where SIGNAL_TYPE = USER, or one where SIGNAL_TYPE = CMD and SIGNAL_SUBTYPE = STOP. **C** The Capture program has completed processing the signal. The Capture program sets this value to C when SIGNAL_TYPE = CMD for all SIGNAL_SUBTYPE values except STOP.
SIGNAL_LSN	**Data type**: CHAR(10) for bit data; **Nullable**: Yes. The log sequence number of the commit record. This value is set only by the Capture program.

On iSeries operating systems, a signal table is associated with each journal used for source tables. These tables are called journal signal tables and have the same structure as the global signal table schema.IBMSNAP_SIGNAL. The name of the journal signal table is *schema*.IBMSNAP_SIGNAL_*xxxx_yyyy*, where *xxxx* is the journal library, and *yyyy* is the journal name. This table is created automatically and is journaled to the source journal on the source server. See "Creating journal signal tables for remote journaling" on page 197 for a description of creating journal signal tables for remote journaling.

schema.IBMSNAP_UOW

Server: Capture control server

Default schema: ASN

Index: IBMSNAP_COMMITSEQ, IBMSNAP_LOGMARKER

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The unit-of-work (UOW) table provides additional information about transactions that have been committed to a source table. For all target table types other than user copy, the Apply program joins the UOW and change data (CD) tables based on matching IBMSNAP_COMMITSEQ values when it applies changes to the target tables. If you cold start the Capture program, all of this table's entries are deleted.

For OS/400: Because Capture for iSeries can start capturing data for a subset of the replication sources, it does not delete all the rows in the UOW table if you do a partial cold start.

The Capture program requires that there is one UOW table for each Capture schema. The Capture program inserts one new row into this table for every log or journal record that is committed at the replication source.

For OS/400: There are some user programs that do not use commitment control. In such cases, Capture for iSeries arbitrarily inserts a new UOW row after a number of rows are written to the CD table. This artificial commitment boundary helps reduce the size of the UOW table.

The Capture program also prunes the UOW table based on information that the Apply program inserts into the prune set (IBMSNAP_PRUNE_SET) table.

For OS/400: The UOW table is pruned by retention limits, not information from the prune set (IBMSNAP_PRUNE_SET) table.

Table 87 provides a brief description of the columns in the UOW table.

Table 87. Columns in the UOW table

Column name	Description
IBMSNAP_UOWID	**Data type**: CHAR(10) for bit data; **Nullable**: No. The unit-of-work identifier from the log record header for this unit of work. You can select that this column be part of a noncomplete CCD target table.
IBMSNAP_COMMITSEQ	**Data type**: CHAR(10) for bit data; **Nullable**: No. The log record sequence number of the captured commit statement. For all target table types other than user copy, the Apply program joins the UOW and CD tables based on the values in this column when it applies changes to the target tables.
IBMSNAP_LOGMARKER	**Data type**: TIMESTAMP; **Nullable**: No. The time (at the Capture control server) that the data was committed.
IBMSNAP_AUTHTKN	**Data type**: VARCHAR(30); **Nullable**: No. The authorization token that is associated with the transaction. This ID is useful for database auditing. For DB2 Universal Database for z/OS, this column is the correlation ID. For DB2 Universal Database for iSeries, this column is the job name of the job that caused a transaction. This column is not automatically copied to other tables; you must select it and copy it as a user data column. You can select that this column be part of a noncomplete CCD target table.
IBMSNAP_AUTHID	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: No. The authorization ID that is associated with the transaction. It is useful for database auditing. For DB2 Universal Database for z/OS, this column is the primary authorization ID. For DB2 Universal Database for iSeries, this column has the name of the user profile ID under which the application that caused the transaction ran. This column holds the ten-character ID padded with blanks. This column is not automatically copied to other tables; you must select it and copy it as a user data column. You can select for this column to be part of a noncomplete CCD target table.

Table 87. Columns in the UOW table (continued)

Column name	Description
IBMSNAP_REJ_CODE	**Data type**: CHAR(1); **Nullable**: No, with default; **Default**: 0. A flag that indicates whether any rows were rejected and rolled back. This value is set only during update-anywhere replication if conflict detection is specified as standard or enhanced when you defined your replication source. You can select that this column be part of a noncomplete CCD target table. **0** No known conflicts occurred in the transaction. **1** A conflict occurred because the same row in the master and replica was updated. The transaction at the replica was rejected and rolled back. **2** The transaction was rejected and rolled back because it was dependent on a prior transaction that was rejected. The prior transaction was rejected because the same row in the master and replica was updated, and the transaction at the replica was rejected and rolled back. **3** The transaction was rejected and rolled back because it contained at least one referential-integrity constraint violation. Because this transaction violates the referential constraints defined on the source table, the Apply program will mark this subscription set as failed. Updates cannot be copied until you correct the referential integrity definitions. **4** The transaction was rejected and rolled back because it was dependent on a prior transaction that was rejected. The prior transaction was rejected because it contained at least one referential-integrity constraint violation.
IBMSNAP_APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No, with default; **Default**: current user name. The Apply qualifier that identifies which Apply program applied the changes. You can select that this column be part of a noncomplete CCD target table.

Tables at the Apply control server and their column descriptions

This section provides greater detail of each table stored at the Apply control server. It also lists and briefly describes the columns in each table. The control tables are listed in alphabetical order, and the columns are listed in the order that they appear in each table from left to right.

ASN.IBMSNAP_APPENQ

Server: Apply control server

Index: APPLY_QUAL

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The Apply enqueue table is used to ensure that only one Apply program is running per Apply qualifier. The Apply program exclusively locks a row in this table until the Apply program is shut down. This table is not used in OS/400.

Table 88 on page 464 provides a brief description of the column in the Apply enqueue table.

Table 88. Column in the Apply enqueue table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: Yes. Uniquely identifies a group of subscription sets that are processed by the same Apply program. This value is case sensitive. You must specify this value when you define a subscription set.

ASN.IBMSNAP_APPLY_JOB (OS/400)

Server: Apply control server

Index: None

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The Apply job table, which is OS/400-specific, is used to guarantee a unique APPLY_QUAL value for all instances of the Apply program running at the Apply control server. A row is added to this table every time an instance of the Apply program is started. If you start a new instance of the Apply program with an APPLY_QUAL value that already exists, the start command fails.

Table 89 provides a brief description of the columns in the Apply job table.

Table 89. Columns in the Apply job table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. A unique identifier for a group of subscription sets. This value is supplied by the user when defining a subscription set. Each instance of the Apply program is started with an APPLY_QUAL value. This value is used during update-anywhere replication to prevent circular replication of the changes made by the Apply program.
CONTROL_SERVER	**Data type**: CHAR(18); **Nullable**: No. The name of the database where the Apply control tables and view are defined.
JOB_NAME	**Data type**: CHAR(10); **Nullable**: No. The fully qualified name of the job that wrote this trace entry: **Position 1–10** APPLY_QUAL **Position 11-20** The ID of the user who started the Apply program **Position 21-26** The job number
USER_NAME	**Data type**: CHAR(10); **Nullable**: No. The name of the user who started a new instance of the Apply program.
JOB_NUMBER	**Data type**: CHAR(6); **Nullable**: No. The job number of the current job for a particular journal. If the journal is not active, this column contains the job number of the last job that was processed.

ASN.IBMSNAP_APPPARMS

Server: Apply control server

Index: APPLY_QUAL

This table contains information that you can update by using SQL.

The Apply parameters table contains parameters that you can modify to control the operations of the Apply program. You can define these parameters to set values such as the name of the Apply control server on which the subscription definitions and Apply program control tables reside. If you make changes to the parameters in this table, the Apply program reads your modifications only during startup.

Table 90 provides a brief description of the columns in the Apply parameters table.

Table 90. Columns in the Apply parameters table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. The Apply qualifier matches the parameters to the Apply program to which these parameters apply.
APPLY_PATH	**Data type**: VARCHAR(1040); **Nullable**: Yes. The location of the work files used by the Apply program. The default is the directory where the program was started.
COPYONCE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N. A flag that indicates whether the Apply program executes one copy cycle for each subscription set that is eligible at the time the Apply program is invoked. **Y** The Apply program executes one copy cycle for each eligible subscription set. **N** The Apply program does not execute one copy cycle for each eligible subscription set.
DELAY	**Data type**: INT; **Nullable**: Yes, with default; **Default:** 6. By default, during continuous replication (that is, when your subscription set uses **sleep**=0 minutes), the Apply program waits 6 seconds after a subscription set is processed successfully before retrying the subscription set. Use a non-zero delay value to save CPU cycles when there is no database activity to be replicated. Use a lower delay value for low latency. **Note:** The **delay** parameter is ignored if **copyonce** is specified.
ERRWAIT	**Data type**: INT; **Nullable**: Yes, with default; **Default:** 300. The number of seconds (1 to 300) that the Apply program waits before retrying after the program encounters an error condition. This parameter is ignored if **copyonce** is specified.
INAMSG	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** Y. A flag that indicates whether the Apply program issues a message when it is inactive. **Y** The Apply program issues a message when inactive. **N** The Apply program does not issue a message when inactive.

Table 90. Columns in the Apply parameters table (continued)

Column name	Description
LOADXIT	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N. A flag that indicates whether the Apply program invokes the IBM-supplied exit routine (ASNLOAD) that uses the export and load utilities to refresh target tables. **Y** The Apply program invokes ASNLOAD. **N** The Apply program does not invoke ASNLOAD.
LOGREUSE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N. A flag that indicates whether the Apply program overwrites the Apply log file or appends to it. **Y** The Apply program reuses the log file by first deleting it and then re-creating it when the Apply program is restarted. **N** The Apply program appends new information to the Apply log file.
LOGSTDOUT	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N. A flag that indicates where the Apply program directs the log file messages: **Y** The Apply program directs log file messages to both the standard out (STDOUT) and the log file. **N** The Apply program directs most log file messages to the log file only. Initialization messages go to both the standard out (STDOUT) and the log file.
NOTIFY	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N. A flag that indicates whether the Apply program should invoke the exit routine (ASNDONE) that returns control to you after the Apply program finishes copying a subscription set. **Y** The Apply program invokes ASNDONE. **N** The Apply program does not invoke ASNDONE.
OPT4ONE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N. A flag that indicates whether the performance of the Apply program is optimized if only one subscription set is defined for the Apply program. **Y** The performance of the Apply program is optimized for one subscription set. **N** The performance of the Apply program is not optimized for one subscription set. This parameter is ignored if **copyonce** is specified.
SLEEP	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** Y. A flag that indicates how the Apply program is to proceed if no new subscription sets are eligible for processing: **Y** The Apply program goes to sleep. **N** The Apply program stops. This parameter is ignored if **copyonce** is specified.

Table 90. Columns in the Apply parameters table (continued)

Column name	Description
SQLERRCONTINUE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N. A flag that indicates whether the Apply program continues processing after it checks the SQLSTATE file for errors. **Y** The Apply program checks the SQLSTATE file for any SQL errors during processing. If an error is found, Apply stops processing. **N** The Apply program does not check the SQLSTATE file and continues processing.
SPILLFILE	**Data type**: VARCHAR(10); **Nullable**: Yes, with default. A flag that indicates where the fetched answer set is stored. **For UNIX and Windows,** valid values are: **disk** (default) A disk file. **For z/OS,** valid values are: **mem** (default) A memory file. **disk** A disk file.
TERM	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** Y. A flag that indicates whether the Apply program terminates when DB2 terminates: **Y** The Apply program terminates when DB2 terminates. **N** The Apply program stays active and waits for DB2 to be restarted. This parameter is ignored if **copyonce** is specified.
TRLREUSE	**Data type**: CHAR(1); **Nullable**: Yes, with default; **Default:** N. A flag that indicates whether the Apply program invokes the IBM-supplied exit routine (ASNLOAD) that uses the export and load utilities to refresh target tables: **Y** The Apply program invokes ASNLOAD. **y** The Apply program does not invoke ASNLOAD.

ASN.IBMSNAP_APPLYTRACE

Server: Apply control server

Index: APPLY_QUAL, TRACE_TIME

The Apply trace table contains important messages from the Apply program. The Apply program does not automatically prune this table, but you can easily automate pruning by adding an SQL statement that runs after one of the subscription sets.

Table 91 on page 468 provides a brief description of the column in the Apply trace table.

IBMSNAP_APPLYTRACE

Table 91. Columns in the Apply trace table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. Uniquely identifies which Apply program inserted the message.
TRACE_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The time at the Apply control server when the row was inserted into this table.
OPERATION	**Data type**: CHAR(8); **Nullable**: No. The type of Apply program operation, for example, initialization, apply, or error condition.
DESCRIPTION	**Data type**: VARCHAR(1024); **Nullable**: No. The message ID followed by the message text. The message ID is the first seven characters of the DESCRIPTION column. The message text starts at the ninth position of the DESCRIPTION column.

ASN.IBMSNAP_APPLYTRAIL

Server: Apply control server

Index: LASTRUN, APPLY_QUAL

The Apply trail table contains audit trail information of all subscription set cycles performed by the Apply program. This table records a history of updates that are performed against subscriptions. It is a repository of diagnostic and performance statistics. The Apply trail table is one of the best places to look if a problem occurs with the Apply program. The Apply program does not automatically prune this table, but you can easily automate pruning by adding an after SQL statement to one of the subscription sets.

Table 92 provides a brief description of the columns in the Apply trail table.

Table 92. Columns in the Apply trail table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. Uniquely identifies which Apply program was processing the subscription set.
SET_NAME	**Data type**: CHAR(18); **Nullable**: No. The name of the subscription set that the Apply program was processing.
SET_TYPE	**Data type**: CHAR(1); **Nullable**: No. The value that appeared in the SET_TYPE column of the subscription set (IBMSNAP_SUBS_SET) table after the most recent Apply cycle. See "ASN.IBMSNAP_SUBS_SET" on page 480 for a description of what each value means.

Table 92. Columns in the Apply trail table (continued)

Column name	Description
WHOS_ON_FIRST	**Data type**: CHAR(1); **Nullable**: No. The following values are used to control the order of processing in update-anywhere replication scenarios. **F** (first) The source table is the replica and the target table is the master. In the case of update conflicts between the replica and the master table, the replica will have its conflicting transactions rejected. F is not used for read-only subscriptions; it is used for update anywhere. **S** (second) The source table is the master table or other source, and the target table is the replica or other copy. In the case of update conflicts between the master and the replica table, the replica will have its conflicting transactions rejected. S is used for all read-only subscriptions.
ASNLOAD	**Data type**: CHAR(1); **Nullable**: Yes. The value used to start the Apply program: **Y** Indicates that the Apply program was started with the parameter **loadxit=y** causing the ASNLOAD user exit routine to be called to perform a full refresh on a subscription set. **N** Indicates that the ASNLOAD exit routine was not called because either a full refresh was not needed or the Apply program was not started with the **loadxit** parameter. **NULL** Indicates that an Apply program error occurred before the Apply program could determine whether the ASNLOAD exit routine should be called.
FULL_REFRESH	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether a full refresh occurred: **Y** Indicates that a full refresh was done for a subscription set. **N** Indicates that a full refresh was not done for a subscription set. **NULL** Indicates that an error occurred before the Apply program could determine whether or not a full refresh was needed.
EFFECTIVE_MEMBERS	**Data type**: INT; **Nullable**: Yes. The number of subscription-set members that are changed during an Apply cycle, either by a full refresh or by the replication of inserts, updates, and deletes. This number ranges between zero and the number of defined subscription-set members.
SET_INSERTED	**Data type**: INT; **Nullable**: No. The total number of rows inserted into subscription-set members during the subscription cycle.
SET_DELETED	**Data type**: INT; **Nullable**: No. The total number of rows deleted from subscription-set members during the subscription cycle.
SET_UPDATED	**Data type**: INT; **Nullable**: No. The total number of rows updated in subscription-set members during the subscription cycle.

Table 92. Columns in the Apply trail table (continued)

Column name	Description
SET_REWORKED	**Data type**: INT; **Nullable**: No. The total number of rows that the Apply program reworked during the last cycle. The Apply program reworks changes under the following conditions: • If an insert fails because the row already exists in the target table, the Apply program converts the insert to an update of the existing row. • If the update fails because the row does not exist in the target table, the Apply program converts the update to an insert.
SET_REJECTED_TRXS	**Data type**: INT; **Nullable**: No. The total number of transactions that were rejected due to an update-anywhere conflict. This column is used only for update-anywhere subscription sets where conflict detection is defined as standard or advanced.
STATUS	**Data type**: SMALLINT; **Nullable**: No. A value that represents the work status for the Apply program after a given cycle: **-1** The replication failed. The Apply program backed out the entire set of rows that it had applied, and no data was committed. If the startup parameter SQLERRCONTINUE = Y, the SQLSTATE that is returned to the Apply program during the last cycle is *not* one of the acceptable errors you indicated in the input file for SQLERRCONTINUE (*apply qualifier*.SQS). **0** The Apply program processed the subscription set successfully. If the startup parameter SQLERRCONTINUE = Y, the Apply program did not encounter any SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and did not reject any rows. **2** The Apply program is processing the subscription set in multiple cycles. It successfully processed a single logical subscription that was divided according to the MAX_SYNCH_MINUTES control column. **16** The Apply program processed the subscription set successfully and returned a status of 0; however, it encountered some SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and rejected some of the rows. See the *apply_qualifier*.ERR file for details about the rows that failed. **Example**: You set SQLERRCONTINUE = Y and indicate that the allowable SQL state is 23502 (SQL code −407). A 23502 error occurs, but no other errors occur. The Apply program finishes processing the subscription set, and it sets the status to 16. On the next execution, a 23502 error occurs, but then a 07006 (SQL code −301) occurs. Now the Apply program stops processing the subscription set, backs out the entire set of rows it had applied, and sets the status to −1 (because no data was committed). **18** The Apply program is processing the subscription set in multiple cycles and returned a status of 2, which means that it successfully processed a single logical subscription that was divided according to the MAX_SYNCH_MINUTES control column. However, it encountered some SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and rejected some of the rows. See the *apply_qualifier*.ERR file for details about the rows that failed.

Table 92. Columns in the Apply trail table (continued)

Column name	Description
LASTRUN	**Data type**: TIMESTAMP; **Nullable**: No. The estimated time that the last subscription began. The Apply program sets the LASTRUN value each time a subscription set is processed. It is the approximate time at the Apply control server that the Apply program begins processing the subscription set.
LASTSUCCESS	**Data type**: TIMESTAMP; **Nullable**: Yes. The Apply control server timestamp for the beginning of the last successful processing of a subscription set.
SYNCHPOINT	**Data type**: CHAR(10) for bit data; **Nullable**: Yes. The Apply program uses this column to record its progress, indicating that it has processed data up to this synchpoint value for the subscription set.
SYNCHTIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The Apply program uses this column to record its progress, indicating that it has processed data up to this timestamp for the subscription set.
SOURCE_SERVER	**Data type**: CHAR(18); **Nullable**: No. The DB2 Universal Database database name where the source tables and views are defined.
SOURCE_ALIAS	**Data type**: CHAR(8); **Nullable**: Yes. The DB2 Universal Database alias corresponding to the source server named in the SOURCE_SERVER column.
SOURCE_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: Yes. The high-level qualifier of the source table or view that the Apply program was processing. This value is set only when the Apply cycle fails.
SOURCE_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: Yes. The name of the source table or view that the Apply program was processing. This value is set only when the Apply cycle fails.
SOURCE_VIEW_QUAL	**Data type**: SMALLINT; **Nullable**: Yes. The value of the source view qualifier for the source table or view that the Apply program was processing. This value is set only when the Apply cycle fails.
TARGET_SERVER	**Data type**: CHAR(18); **Nullable**: No. The database name of the server where target tables or views are stored.
TARGET_ALIAS	**Data type**: CHAR(8); **Nullable**: Yes. The DB2 Universal Database alias corresponding to the target server named in the TARGET_SERVER column.
TARGET_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: No. The high-level qualifier of the target table that the Apply program was processing. This value is set only when the Apply cycle fails.

Table 92. Columns in the Apply trail table (continued)

Column name	Description
TARGET_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No. The name of the target table that the Apply program was processing. This value is set only when the Apply cycle fails.
CAPTURE_SCHEMA	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: No. The schema name of the Capture server tables for this subscription set.
TGT_CAPTURE_SCHEMA	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: Yes. If the target table is also the source for another subscription set (such as an external CCD table in a multi-tier configuration or a replica table in an update-anywhere configuration), this column contains the Capture schema that will be used when the table is acting as a source.
FEDERATED_SRC_SRVR	**Data type**: VARCHAR(18); **Nullable**: Yes. The name of the federated remote server that is the source for the subscription set, which applies only to non-DB2 relational sources.
FEDERATED_TGT_SRVR	**Data type**: VARCHAR(18); **Nullable**: Yes. The name of the federated remote server that is the target for the subscription set, which applies only to non-DB2 relational target servers.
JRN_LIB	**Data type**: CHAR(10); **Nullable**: Yes. This column, which applies only to OS/400 Capture servers, is the library name of the journal that the source table uses.
JRN_NAME	**Data type**: CHAR(10); **Nullable**: Yes. This column, which applies only to OS/400 Capture servers, is the name of the journal used by a source table. An asterisk followed by nine blanks in this column means that the source table is currently not in a journal, in which case it is not possible to capture data for this source table.
COMMIT_COUNT	**Data type**: SMALLINT; **Nullable**: Yes. The value of the COMMIT_COUNT from the last Apply cycle, which is recorded in the subscription sets (IBMSNAP_SUBS_SET) table.
OPTION_FLAGS	**Data type**: CHAR(4); **Nullable**: No. Reserved for future options of DB2 replication. Currently this column contains the default value of NNNN.
EVENT_NAME	**Data type**: CHAR(18); **Nullable**: Yes. A unique character string used to represent the event that triggered the set to be processed.
ENDTIME	**Data type**: TIMESTAMP; **Nullable**: No, with default; **Default**: current timestamp. The timestamp at the Apply control server when the Apply program finished processing the subscription set. To find out how long a set took to process, subtract LASTRUN from ENDTIME.
SOURCE_CONN_TIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The timestamp at the Capture control server when the Apply program first connects to fetch source data.

Table 92. Columns in the Apply trail table (continued)

Column name	Description
SQLSTATE	**Data type**: CHAR(5); **Nullable**: Yes. The SQL state code for a failed execution. Otherwise, NULL.
SQLCODE	**Data type**: INT; **Nullable**: Yes. The SQL error code for a failed execution. Otherwise, NULL.
SQLERRP	**Data type**: CHAR(8); **Nullable**: Yes. The database product identifier of the server where an SQL error occurred that caused a failed execution. Otherwise, NULL.
SQLERRM	**Data type**: VARCHAR(70); **Nullable**: Yes. The SQL error information for a failed execution.
APPERRM	**Data type**: VARCHAR(760); **Nullable**: Yes. The Apply error message ID and text for a failed execution.

ASN.IBMSNAP_SUBS_COLS

Server: Apply control server

Index: APPLY_QUAL, SET_NAME, WHOS_ON_FIRST, TARGET_OWNER, TARGET_TABLE, TARGET_NAME

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The subscription columns table contains information about the columns of the subscription-set members that are copied in a subscription set. Rows are automatically inserted into or deleted from this table when information changes in one or more columns of a source and target table pair.

Use this table if you need information about specific columns in a subscription-set member.

Table 93 provides a brief description of the columns in the subscription columns table.

Table 93. Columns in the subscription columns table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. Uniquely identifies which Apply program processes this subscription-set member.
SET_NAME	**Data type**: CHAR(18); **Nullable**: No. The name of a subscription set that this member belongs to.

Table 93. Columns in the subscription columns table (continued)

Column name	Description
WHOS_ON_FIRST	**Data type**: CHAR(1); **Nullable**: No. The following values are used to control the order of processing in update-anywhere replication scenarios. **F** (first) The source table is the replica and the target table is the master. In the case of update conflicts between the replica and the master table, the replica will have its conflicting transactions rejected. F is not used for read-only subscriptions; it is used for update anywhere. **S** (second) The source table is the master table or other source, and the target table is the replica or other copy. In the case of update conflicts between the master and the replica table, the replica will have its conflicting transactions rejected. S is used for all read-only subscriptions.
TARGET_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: No. The high-level qualifier for a target table or view.
TARGET_TABLE	**Data type**: VARCHAR(128); VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No. The table or view to which data is being applied.
COL_TYPE	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates the type of column: **A** An after-image column. **B** A before-image column. **C** A computed column or an SQL expression using scalar functions. **D** A DATALINK column. **F** A computed column using column functions. **L** A LOB indicator value. **P** A before-image predicate column. **R** A relative record number column, provided by the system and used as a primary key column. Used only by DB2 DataPropagator for iSeries.
TARGET_NAME	**Data type**: VARCHAR(30); **Nullable**: No. The name of the target table or view column. It does not need to match the source column name. Internal-CCD column names cannot be renamed. They must match the source-table column names.
IS_KEY	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates whether the column is part of the target key, which can be either a unique index or primary key of a condensed target table: **Y** The column is all or part of the target key. **N** The column is not part of the target key.
COLNO	**Data type**: SMALLINT; **Nullable**: No. The numeric location of the column in the original source, to be preserved relative to other user columns in displays and subscriptions.

Table 93. Columns in the subscription columns table (continued)

Column name	Description
EXPRESSION	**Data type**: VARCHAR(254); **Nullable**: No. The source column name or an SQL expression used to create the target column contents.

ASN.IBMSNAP_SUBS_EVENT

Server: Apply control server

Index: EVENT_NAME, EVENT_TIME

This table contains information that you can update using SQL.

The subscription events table contains information about the event triggers that are associated with a subscription set. It also contains names and timestamps that are associated with the event names. You insert a row into this table when you create a new event to start an Apply program. See "Event-based scheduling" on page 66.

Table 94 provides a brief description of the columns in the subscription events table.

Table 94. Columns in the subscription events table

Column name	Description
EVENT_NAME	**Data type**: CHAR(18); **Nullable**: No. The unique identifier of an event. This identifier is used to trigger replication for a subscription set.
EVENT_TIME	**Data type**: TIMESTAMP; **Nullable**: No. An Apply control server timestamp of a current or future posting time. User applications that signal replication events provide the values in this column.
END_SYNCHPOINT	**Data type**: CHAR(10) for bit data; **Nullable**: Yes. A log sequence number that tells the Apply program to apply only data that has been captured up to this point. You can find the exact END_SYNCHPOINT that you want to use by referring to the signal table and finding the precise log sequence number associated with a timestamp. Any transactions that are committed beyond this point in the log are not replicated until a later event is posted. If you supply values for END_SYNCHPOINT and END_OF_PERIOD, the Apply program uses the END_SYNCHPOINT value because it then does not need to perform any calculations from the control tables to find the maximum log sequence number to replicate.
END_OF_PERIOD	**Data type**: TIMESTAMP; **Nullable**: Yes. A timestamp used by the Apply program, which applies only data that has been logged up to this point. Any transactions that are committed beyond this point in the log are not replicated until a later event is posted.

ASN.IBMSNAP_SUBS_MEMBR

Server: Apply control server

Index: APPLY_QUAL, SET_NAME, WHOS_ON_FIRST, SOURCE_OWNER, SOURCE_TARGET, SOURCE_VIEW_QUAL, TARGET_OWNER, TARGET_TABLE

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The subscription members table contains information about the individual source and target table pairs defined for a subscription set. A single row is automatically inserted into this table when you add a subscription set member.

Use this table to identify a specific source and target table pair within a subscription set.

Table 95 provides a brief description of the columns in the subscription member table.

Table 95. Columns in the subscription member table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. Uniquely identifies which Apply program processes this subscription-set member.
SET_NAME	**Data type**: CHAR(18); **Nullable**: No. The name of the subscription set that this member belongs to.
WHOS_ON_FIRST	**Data type**: CHAR(1); **Nullable**: No. The following values are used to control the order of processing in update-anywhere replication scenarios. **F** (first) The source table is the replica and the target table is the master. In the case of update conflicts between the replica and the master table, the replica will have its conflicting transactions rejected. F is not used for read-only subscriptions; it is used for update anywhere. **S** (second) The source table is the master table or other source, and the target table is the replica or other copy. In the case of update conflicts between the master and the replica table, the replica will have its conflicting transactions rejected. S is used for all read-only subscriptions.
SOURCE_OWNER	**Data type**: VARCHAR(30); VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: No. The high-level qualifier for the source table or view for this member.
SOURCE_TABLE	**Data type**: VARCHAR(128); VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No. The name of the source table or view for this member.
SOURCE_VIEW_QUAL	**Data type**: SMALLINT; **Nullable**: No. Supports the view of physical tables by matching the similar column in the register table. This value is set to 0 for physical tables that are defined as sources and is greater than 0 for views that are defined as sources. This column is used to support multiple subscriptions for different source views with identical SOURCE_OWNER and SOURCE_TABLE column values.
TARGET_OWNER	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: No. The high-level qualifier for the target table or view for this member.

Table 95. Columns in the subscription member table (continued)

Column name	Description
TARGET_TABLE	**Data type**: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; **Nullable**: No. The name of the target table or view for this member.
TARGET_CONDENSED	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates: **Y** For any given primary key value, the target table shows only one row. **N** All changes must remain to retain a complete update history. **A** The target table is a base aggregate or change aggregate tables.
TARGET_COMPLETE	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates: **Y** The target table contains a row for every primary key value of interest. **N** The target table contains some subset of rows of primary key values.
TARGET_STRUCTURE	**Data type**: SMALLINT; **Nullable**: No. The structure of the target table: **1** User table **3** CCD table **4** Point-in-time table **5** Base aggregate table **6** Change aggregate table **7** Replica **8** User copy
PREDICATES	**Data type**: VARCHAR(1024); **Nullable**: Yes. Lists the predicates to be placed in a WHERE clause for the table in the TARGET_TABLE column. This WHERE clause creates a row subset of the source table. Predicates are recognized only when WHOS_ON_FIRST is set to S. The predicate cannot contain an ORDER BY clause because the Apply program cannot generate an ORDER BY clause. Aggregate tables require a dummy predicate followed by a GROUP BY clause. Because the Apply program uses these predicates for both full-refresh and change-capture replication, this column cannot contain predicates that involve columns in the CD or UOW table. Predicates that contain CD or UOW table references are stored in the UOW_CD_PREDICATES column.

Table 95. Columns in the subscription member table (continued)

Column name	Description
MEMBER_STATE	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates what state the member is in: **N** (New) The member is new to this subscription set. Also, any members that were recently enabled will appear in this state. **L** (Loaded) The members of this subscription set have been loaded, but there has not yet been a change capture cycle. **S** (Synchronized) The member has been advanced from the new (N) state to the loaded (L) state, and is now synchronized with all the other subscription-set members that are in the synchronized state. When all members of a subscription set are in the synchronized state, change replication can occur at the subscription set level. **D** (Disabled) The member is disabled for this subscription set.
TARGET_KEY_CHG	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates how the Apply program handles updates when, at the source table, you change the source columns for the target key columns of a target table: **Y** The Apply program updates the target table based on the before images of the target key column, meaning that the Apply program changes the predicate to the old values instead of the new. Make sure you have registered each before-image column of the target key so it is present in the CD table. For the corresponding registration entry in the register table, make sure the value in the CHG_UPD_TO_DEL_INS column is set to N. **N** The Apply program uses logic while processing updates and deletes that assume that the columns that make up the target key are never updated.
UOW_CD_PREDICATES	**Data type**: VARCHAR(1024); **Nullable**: Yes. Contains predicates that include columns from the CD or UOW table that the Apply program needs only for change-capture replication, and not for full refreshes. During change-capture replication, the Apply program processes the predicates in this column and those in the PREDICATES column. During a full refresh, the Apply program processes only the predicates in the PREDICATES column.

Table 95. Columns in the subscription member table (continued)

Column name	Description
JOIN_UOW_CD	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the Apply program does a join of the CD and UOW tables when processing a user copy target table. This flag is needed when you define a subscription-set member with predicates that use columns from the UOW table that are not in the CD table. If the target table type is anything except user copy, then the Apply program uses a join of the CD and UOW tables when processing the member, and it ignores this column when processing the member. **Y** The Apply program uses a join of the CD and UOW tables when processing the member. **N** The Apply program does not use a join of the CD and UOW tables when processing the member; it reads changes only from the CD table. **NULL** The Apply program ignores this column when processing the member. If the target table is a user copy and the value in this column is null, then the Apply program does not do a join of the CD and UOW tables when processing the member.
LOADX_TYPE	**Data type**: SMALLINT; **Nullable**: Yes. The type of load for this member. The value in this column is used to override the defaults. **NULL** **For z/OS**: The cross loader function (available with the DB2 Utilities Suite) is used for this member. **For UNIX and Windows**: The ASNLOAD exit determines the most appropriate utility for this member (option 3, 4, or 5). **1** ASNLOAD is not used for this member. This effectively turns ASNLOAD option off for a particular subscription-set member even if you specified LOADX on startup. **2** A user-defined or user-modified ASNLOAD exit code is used. **3** The cross loader function (available with the DB2 Utilities Suite) is used for this member. **4** **For UNIX and Windows only**: EXPORT/LOAD is used for this member. **5** **For UNIX and Windows only**: EXPORT/IMPORT is used for this member. **Restriction for UNIX and Windows**: The LOAD utility is not supported for range-clustered tables. To do a full refresh of a range-clustered table, you can either use the DB2 IMPORT utility or the Apply program to do a full refresh of the table through SQL.
LOADX_SRC_N_OWNER	**Data type**: VARCHAR(30); **Nullable**: Yes. The user-created nickname owner. This value is required when all of the following conditions exist: • The cross loader function is used for this member (LOADX_TYPE is 3) • The target server is UNIX or Windows • The source is not a nickname

Table 95. Columns in the subscription member table (continued)

Column name	Description
LOADX_SRC_N_TABLE	**Data type**: VARCHAR(128); **Nullable**: Yes. The user-created nickname table. This value is required when all of the following conditions exist: • The cross loader function is used for this member (LOADX_TYPE is 3) • The target server is UNIX or Windows • The source is not a nickname

ASN.IBMSNAP_SUBS_SET

Server: Apply control server

Index: APPLY_QUAL, SET_NAME, WHOS_ON_FIRST

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data.

The subscription sets table lists all of the subscription sets that are defined at the Apply control server and documents the replication progress for these sets. Rows are inserted into this table when you create your subscription set definition.

Table 96 provides a brief description of the columns in the subscription sets table.

Table 96. Columns in the subscription sets table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. Uniquely identifies which Apply program processes this subscription set.
SET_NAME	**Data type**: CHAR(18); **Nullable**: No. The name of the subscription set.
SET_TYPE	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates whether the set is read only or read/write: **R** The set is read only. **U** The set is an update-anywhere configuration, and therefore is read/write.
WHOS_ON_FIRST	**Data type**: CHAR(1); **Nullable**: No. The following values are used to control the order of processing in update-anywhere replication scenarios. **F** (first) The source table is the replica and the target table is the master. In the case of update conflicts between the replica and the master table, the replica will have its conflicting transactions rejected. F is not used for read-only subscriptions; it is used for update anywhere. **S** (second) The source table is the master table or other source, and the target table is the replica or other copy. In the case of update conflicts between the master and the replica table, the replica will have its conflicting transactions rejected. S is used for all read-only subscriptions.

Table 96. Columns in the subscription sets table (continued)

Column name	Description
ACTIVATE	**Data type**: SMALLINT; **Nullable**: No. A flag that indicates whether the Apply program will process the set during its next cycle: **0** The subscription set is deactivated. The Apply program will not process the set. **1** The subscription set is active indefinitely. The Apply program will process the set during each Apply cycle until you deactivate the set or until the Apply program is unable to process it. **2** The subscription set is active for only one Apply cycle. The Apply program will process the set once and then deactivate the set.
SOURCE_SERVER	**Data type**: CHAR(18); **Nullable**: No. The database name of the Capture control server where the source tables and views are defined.
SOURCE_ALIAS	**Data type**: CHAR(8); **Nullable**: Yes. The DB2 Universal Database alias corresponding to the Capture control server that is named in the SOURCE_SERVER column.
TARGET_SERVER	**Data type**: CHAR(18); **Nullable**: No. The database name of the server where target tables or views are stored.
TARGET_ALIAS	**Data type**: CHAR(8); **Nullable**: Yes. The DB2 Universal Database alias corresponding to the target server named in the TARGET_SERVER column.

Table 96. Columns in the subscription sets table (continued)

Column name	Description
STATUS	**Data type**: SMALLINT; **Nullable**: No. A value that represents the work status for the Apply program after a given cycle: **-1** The replication failed. The Apply program backed out the entire set of rows it had applied, and no data was committed. If the startup parameter SQLERRCONTINUE = Y, the SQLSTATE that is returned to the Apply program during the last cycle is *not* one of the acceptable errors you indicated in the input file for SQLERRCONTINUE (*apply qualifier*.SQS). **0** The Apply program processed the subscription set successfully. If the startup parameter SQLERRCONTINUE = Y, the Apply program did not encounter any SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and did not reject any rows. **2** The Apply program is processing the subscription set in multiple cycles. It successfully processed a single logical subscription that was divided according to the MAX_SYNCH_MINUTES control column. **16** The Apply program processed the subscription set successfully and returned a status of 0; however, it encountered some SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and rejected some of the rows. See the *apply_qualifier*.ERR file for details about the rows that failed. **Example**: You set SQLERRCONTINUE = Y and indicate that the allowable SQL state is 23502 (SQL code −407). A 23502 error occurs, but no other errors occur. The Apply program finishes processing the subscription set, and it sets the status to 16. On the next execution, a 23502 error occurs, but then a 07006 (SQL code −301) occurs. Now the Apply program stops processing the subscription set, backs out the entire set of rows it had applied, and sets the status to −1 (because no data was committed). **18** The Apply program is processing the subscription set in multiple cycles and returned a status of 2, meaning that it successfully processed a single logical subscription that was divided according to the MAX_SYNCH_MINUTES control column. However, it encountered some SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in *apply_qualifier*.SQS) and rejected some of the rows. See the *apply_qualifier*.ERR file for details about the rows that failed.
LASTRUN	**Data type**: TIMESTAMP; **Nullable**: No. The estimated time that the last subscription set began. The Apply program sets the LASTRUN value each time a subscription set is processed. It is the approximate time at the Apply control server when the Apply program begins processing the subscription set.

Table 96. Columns in the subscription sets table (continued)

Column name	Description
REFRESH_TYPE	**Data type**: CHAR(1); **Nullable**: No. The type of scheduling that is used to prompt the Apply program to process this subscription set: **R** The Apply program uses time-based scheduling. It uses the value in SLEEP_MINUTES to determine when to start processing the subscription set. **E** The Apply program uses event-based scheduling. It checks the time value in the subscription events (IBMSNAP_SUBS_EVENT) table to determine when to start processing the subscription set. Before any replication (change capture or full refresh) can begin, an event must occur. **B** The Apply program uses both time-based and event-based scheduling. Therefore, it processes the subscription set based on either the time or event criteria.
SLEEP_MINUTES	**Data type**: INT; **Nullable**: Yes. Specifies the time (in minutes) of inactivity between subscription set processing. The processing time is used only when REFRESH_TYPE is R or B. If the value of SLEEP_MINUTES is NULL, the Apply program will process the set continuously. The Apply program will process the set as often as possible, but will also process all other active subscription sets with the same Apply qualifier.
EVENT_NAME	**Data type**: CHAR(18); **Nullable**: Yes. A unique character string used to represent the name of an event. Use this identifier to update the subscription events table when you want to trigger replication for a subscription set. The event name is used only when REFRESH_TYPE is E or B.
LASTSUCCESS	**Data type**: TIMESTAMP; **Nullable**: Yes. The Apply control server timestamp for the beginning of the last successful processing of a subscription set.
SYNCHPOINT	**Data type**: CHAR(10) for bit data; **Nullable**: Yes. The Apply program uses this column to record its progress, indicating that it has processed data up to this synchpoint value for the subscription set.
SYNCHTIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The Apply program uses this column to record its progress, indicating that it has processed data up to this timestamp for the subscription set.
CAPTURE_SCHEMA	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: No. The schema name of the Capture control tables that process the source for this subscription set.
TGT_CAPTURE_SCHEMA	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: Yes. If the target table is also the source for another subscription set (such as an external CCD table in a multi-tier configuration or a replica table in an update-anywhere configuration), then this column contains the Capture schema that is used when the table is acting as a source.

Table 96. Columns in the subscription sets table (continued)

Column name	Description
FEDERATED_SRC_SRVR	**Data type**: VARCHAR(18); **Nullable**: Yes. The name of the federated remote server that is the source for the subscription set, which applies only to non-DB2 relational sources.
FEDERATED_TGT_SRVR	**Data type**: VARCHAR(18); **Nullable**: Yes. The name of the federated remote server that is the target for the subscription set, which applies only to non-DB2 relational targets.
JRN_LIB	**Data type**: CHAR(10); **Nullable**: Yes. This column, which applies only to OS/400 Capture servers, is the library name of the journal that the source table uses.
JRN_NAME	**Data type**: CHAR(10); **Nullable**: Yes. This column, which applies only to OS/400 Capture servers, is the name of the journal used by a source table. An asterisk followed by nine blanks in this column means that the source table is currently not in a journal, in which case it is not possible to capture data for this source table.
OPTION_FLAGS	**Data type**: CHAR(4); **Nullable**: No. Reserved for future options of DB2 replication. Currently this column contains the default value of NNNN.
COMMIT_COUNT	**Data type**: SMALLINT; **Nullable**: Yes. A flag that indicates the type of processing that the Apply program performs for a subscription set: **NULL** This is the default setting for a read-only subscription set. The Apply program will process fetched answer sets for the *n* subscription-set members one member at a time, until all data has been processed, and then will issue a single commit at the end of the data processing for the whole set. The advantage of using this COMMIT_COUNT setting is that the processing might complete faster. *Integer not NULL* The Apply program processes the subscription set in a transactional mode. After all answer sets are fetched, the contents of the spill files will be applied in the order of commit sequence, ordering each transaction by the IBMSNAP_INTENTSEQ value order. This type of processing allows all spill files to be open and processed at the same time. A commit will be issued following the number of transactions specified in this column. For example, 1 means commit after each transaction, 2 means commit after each two transactions, and so on. An integer of 0 means that a single commit will be issued after all fetched data is applied. The advantage to using transactional mode processing is that the processing allows for referential integrity constraints at the target, and interim commits can be issued.
MAX_SYNCH_MINUTES	**Data type**: SMALLINT; **Nullable**: Yes. A time-threshold limit to regulate the amount of change data to fetch and apply during a subscription cycle. The Apply program breaks the subscription set processing into mini-cycles based on the IBMSNAP_LOGMARKER column in the UOW or CCD table at the Capture server and issues a COMMIT at the target server after each successful mini-cycle. The limit is automatically recalculated if the Apply program encounters a resource constraint that makes the set limit unfeasible. MAX_SYNCH_MINUTES values that are less than 1 will be treated the same as a MAX_SYNCH_MINUTES value equal to null.

Table 96. Columns in the subscription sets table (continued)

Column name	Description
AUX_STMTS	**Data type**: SMALLINT; **Nullable**: No. The number of SQL statements that you define in the subscription statement (IBMSNAP_SUBS_STMTS) table that can run before or after the Apply program processes a subscription set.
ARCH_LEVEL	**Data type**: CHAR(4); **Nullable**: No. The architectural level of the replication control tables. This column identifies the rules under which a row was created. This level is defined by IBM, and for Version 8 **0801** Version 8 SQL Replication **0803** Version 8 SQL Replication with enhanced support for Oracle sources **0805** Version 8 SQL Replication with support for DB2 for z/OS new-function mode

ASN.IBMSNAP_SUBS_STMTS

Server: Apply control server

Index: APPLY_QUAL, SET_NAME, WHOS_ON_FIRST, BEFORE_OR_AFTER, STMT_NUMBER

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause unexpected results and loss of data. The number of entries for a subscription should be reflected in the ASN.IBMSNAP_SUBS_SET.AUX_STMTS column. If AUX_STMTS is zero for a subscription set, the corresponding entries in the subscription statements table are ignored by the Apply program.

The subscription statements table contains the user-defined SQL statements or stored procedure calls that will be executed before or after each subscription-set processing cycle. Execute immediately (EI) statements or stored procedures can be executed at the source or target server only.

This table is populated when you define a subscription set that uses SQL statements or stored procedure calls.

Table 97 provides a brief description of the columns in the subscription statements table.

Table 97. Columns in the subscription statements table

Column name	Description
APPLY_QUAL	**Data type**: CHAR(18); **Nullable**: No. Uniquely identifies which Apply program processes the SQL statement or stored procedure.
SET_NAME	**Data type**: CHAR(18); **Nullable**: No. The name of the subscription set that the SQL statement or stored procedure is associated with.

Table 97. Columns in the subscription statements table (continued)

Column name	Description
WHOS_ON_FIRST	**Data type**: CHAR(1); **Nullable**: No. The following values are used to control the order of processing in update-anywhere replication scenarios. **F** (first) The target table is the user table or parent replica. The source table is the dependent replica and, in the case of update conflicts between the source table and the target table, the source table will have its conflicting transactions rejected. F is not used for read-only subscriptions. **S** (second) The source table is the user table, parent replica, or other source. The target table is the dependent replica or other copy and, in the case of update conflicts between the source table and the target table, the target table will have its conflicting transactions rejected. S is used for all read-only subscriptions.
BEFORE_OR_AFTER	**Data type**: CHAR(1); **Nullable**: No. A value that indicates when and where the statement is issued: **A** The statement is executed at the target server after all of the answer-set rows are applied. **B** The statement is executed at the target server before any of the answer-set rows are applied. **S** The statement is executed at the Capture control server before opening the answer-set cursors. **G** Reserved for use by DB2 replication. **X** Reserved for use by DB2 replication.
STMT_NUMBER	**Data type**: SMALLINT; **Nullable**: No. Defines the relative order of execution within the scope of the BEFORE_OR_AFTER column value.
EI_OR_CALL	**Data type**: CHAR(1); **Nullable**: No. A value that indicates: **E** The SQL statement should be run as an EXEC SQL EXECUTE IMMEDIATE. **C** The SQL statement contains a stored procedure name to run as an EXEC SQL CALL.
SQL_STMT	**Data type**: VARCHAR(1024); **Nullable**: Yes. One of the following values: **Statement** The SQL statement should run as an EXEC SQL EXECUTE IMMEDIATE statement if EI_OR_CALL is E. **Procedure** The eight-byte name of an SQL stored procedure, without parameters, or the CALL keyword that runs as an EXEC SQL CALL statement if EI_OR_CALL is C.

Table 97. Columns in the subscription statements table (continued)

Column name	Description
ACCEPT_SQLSTATES	**Data type**: VARCHAR(50); **Nullable**: Yes. One to ten five-byte SQLSTATE values that you specified when you defined the subscription set. These non-zero values are accepted by the Apply program as a successful execution. Any other values will cause a failed execution.

Tables at the Monitor control server and their column descriptions

This section provides greater detail of each table stored at the Monitor control server. It also lists and briefly describes the columns in each table. The control tables are listed in alphabetical order, and the columns are listed in the order that they appear in each table from left to right.

IBMSNAP_ALERTS table

Server: Monitor control server

Index: MONITOR_QUAL, COMPONENT, SERVER_NAME, SCHEMA_OR_QUAL, SET_NAME, CONDITION_NAME, ALERT_CODE

The IBMSNAP_ALERTS table contains a record of all the alerts issued by the Replication Alert Monitor. The table records what alert conditions occur, at which servers they occur, and when they were detected.

Table 98 provides a brief description of the columns in the IBMSNAP_ALERTS table.

Table 98. Columns in the IBMSNAP_ALERTS table

Column name	Description
MONITOR_QUAL	**Data type:** CHAR(18); **Nullable:** No. The Monitor qualifier that identifies which Replication Alert Monitor program issued the alert.
COMPONENT	**Data type:** CHAR(1); **Nullable:** No. The replication component that is being monitored: **C** Capture program **A** Apply program **S** Q Capture program **R** Q Apply program
SERVER_NAME	**Data type:** CHAR(18); **Nullable:** No. The name of the Capture control server, Apply control server, Q Capture server, or Q Apply server where the alert condition occurred.
SERVER_ALIAS	**Data type:** CHAR(8); **Nullable:** Yes. The DB2 UDB alias of the Capture control server, Apply control server, Q Capture server, or Q Apply server where the alert condition occurred.

Table 98. Columns in the IBMSNAP_ALERTS table (continued)

Column name	Description
SCHEMA_OR_QUAL	**Data type:** VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable:** No. The Capture schema, Apply schema, Q Capture schema, or Q Apply schema that is being monitored.
SET_NAME	**Data type:** CHAR(18); **Nullable:** No, with default; **Default:** Current subscription set. If you set an alert condition for the Apply program, this column specifies the name of the subscription set that is being monitored. If you do not specify a set name, then monitoring is done at the Apply-qualifier level, meaning that every set within the given Apply qualifier is monitored. If you set an alert condition for the Q Apply receive queue depth or spill queue depth, this column specifies the name of the receive queue or spill queue that is being monitored.
CONDITION_NAME	**Data type:** CHAR(18); **Nullable:** No. The condition code that was tested when the alert was triggered.
OCCURRED_TIME	**Data type:** TIMESTAMP; **Nullable:** No. The time that the alert condition occurred at the Capture control server, Apply control server, Q Capture server, or Q Apply server.
ALERT_COUNTER	**Data type:** SMALLINT; **Nullable:** No. The number of times that this alert has been previously detected in consecutive monitor cycles.
ALERT_CODE	**Data type:** CHAR(10); **Nullable:** No. The message code that was issued when the alert occurred.
RETURN_CODE	**Data type:** INT; **Nullable:** No. The integer value returned by a user condition.
NOTIFICATION_SENT	**Data type:** CHAR(1); **Nullable:** No. A flag that indicates whether a notification message was sent: **Y** A notification message was sent. **E** A notification was not sent because the **email_server** parameter was not specified. **N** A notification was not sent because the number of notifications already reached the limit set by the **max_notifications_per_alert** parameter.
ALERT_MESSAGE	**Data type:** VARCHAR(1024); **Nullable:** No. The text of the message that was sent, including the message code.

IBMSNAP_CONDITIONS table

Server: Monitor control server

Index: MONITOR_QUAL, COMPONENT, SERVER_NAME, SCHEMA_OR_QUAL, SET_NAME, CONDITION_NAME

The IBMSNAP_CONDITIONS table contains the alert conditions for which the Replication Alert Monitor will contact someone, and it contains the group or individual's name to contact if a particular condition occurs. The Replication Alert Monitor can monitor a combination of conditions on Capture control servers, Apply control servers, Q Capture servers, and Q Apply servers.

Table 99 provides a brief description of the columns in the IBMSNAP_CONDITIONS table.

Table 99. Columns in the IBMSNAP_CONDITIONS table

Column name	Description
SERVER_NAME	**Data type**: CHAR(18); **Nullable**: No. The name of the Capture control server, Apply control server, Q Capture server, or Q Apply server where this condition is being monitored.
COMPONENT	**Data type**: CHAR(1); **Nullable**: No. The replication component that is being monitored: **C** Capture program **A** Apply program **S** Q Capture program **R** Q Apply program
SCHEMA_OR_QUAL	**Data type**: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; **Nullable**: No. The Capture schema, Apply schema, Q Capture schema, or Q Apply schema that is being monitored.
SET_NAME	**Data type**: CHAR(18); **Nullable**: No; **Default**: Current subscription set. If you set an alert condition for the Apply program, this column specifies the name of the subscription set that is being monitored. If you do not specify a set name, then monitoring is done at the Apply-qualifier level, meaning that every set within the given Apply qualifier is monitored. If you set an alert condition for the Q Apply receive queue depth or spill queue depth, this column specifies the name of the receive queue or spill queue that is being monitored.
MONITOR_QUAL	**Data type**: CHAR(18); **Nullable**: No. The Monitor qualifier that identifies which Replication Alert Monitor program is monitoring the Capture control server, Apply control server, Q Capture server, or Q Apply server for this condition.
SERVER_ALIAS	**Data type**: CHAR(8); **Nullable**: Yes. The DB2 UDB alias of the Capture control server, Apply control server, Q Capture server, or Q Apply server where this condition is being monitored..
ENABLED	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates whether the Replication Alert Monitor will process this condition during the next monitoring cycle: **Y** The Replication Alert Monitor will process this definition during the next cycle. **N** The Replication Alert Monitor will ignore this definition during the next cycle.

Table 99. Columns in the IBMSNAP_CONDITIONS table (continued)

Column name	Description
CONDITION_NAME	**Data type**: CHAR(18); **Nullable**: No.The name of the condition that the Replication Alert Monitor is monitoring at the given Capture control server, Apply control server, Q Capture server, or Q Apply server. Conditions for the Capture program begin with CAPTURE. Conditions for the Apply program begin with APPLY. Conditions for the Q Capture program begin with QCAPTURE. Conditions for the Q Apply program begin with QAPPLY. **CAPTURE_STATUS** The status of the Capture program. **CAPTURE_ERRORS** Whether the Capture program posted any error messages. **CAPTURE_WARNINGS** Whether the Capture program posted any warning messages. **CAPTURE_LASTCOMMIT** The last time the Capture program committed data during the last monitor cycle. **CAPTURE_CLATENCY** The Capture program's current latency. **CAPTURE_HLATENCY** Whether the Capture program's latency is greater than a certain number of seconds. **CAPTURE_MEMORY** The amount of memory (in megabytes) that the Capture program is using.
CONDITION_NAME (Continued)	**APPLY_STATUS** The status of the Apply program. **APPLY_SUBSFAILING** Whether any subscription sets failed. **APPLY_SUBSINACT** Whether any subscription sets either failed or are inactive. **APPLY_ERRORS** Whether the Apply program posts any error messages. **APPLY_WARNINGS** Whether the Apply program posts any warning messages. **APPLY_FULLREFRESH** Whether a full refresh occurred. **APPLY_REJTRANS (update anywhere)** Whether the Apply program rejects transactions in any subscription set. **APPLY_SUBSDELAY** Whether the Apply program delays more than the time that you specified in the PARM_INT parameter. **APPLY_REWORKED** Whether the Apply program reworked any rows at the target table. **APPLY_LATENCY** Whether the end-to-end latency of the Apply program exceeds a threshold.

Table 99. Columns in the IBMSNAP_CONDITIONS table (continued)

Column name	Description
CONDITION_NAME (Continued)	**QCAPTURE_STATUS** Whether the Q Capture program is down. **QCAPTURE_ERRORS** Whether the Q Capture program posted any error messages. **QCAPTURE_WARNINGS** Whether the Q Capture program posted any warning messages. **QCAPTURE_LATENCY** Whether the Q Capture latency (the difference between the last insert into the IBMQREP_CAPMON table and the timestamp of the last transaction that the Q Capture program read in the DB2 log) exceeds a threshhold. **QCAPTURE_MEMORY** Whether the memory amount that the Q Capture program used exceeds a threshold. **QCAPTURE_TRANSIZE** Whether a transaction exceeds the MAX_TRANS_SIZE (maximum transaction size) that is set in the IBMQREP_CAPMON table. **QCAPTURE_SUBSINACT** Whether a Q Subscription changed to I (inactive) state.
CONDITION_NAME (Continued)	**QAPPLY_STATUS** Whether the Q Apply program is down. **QAPPLY_ERRORS** Whether the Q Apply program posted any error messages. **QAPPLY_WARNINGS** Whether the Q Apply program posted any warning messages. **QAPPLY_LATENCY** Whether the queue latency (the time it takes for a message to go from the send queue to the receive queue) exceeds a threshold. **QAPPLY_EELATENCY** Whether the end-to-end latency (the time it takes for a transaction to replicate from the source to the target) exceeds a threshold. **QAPPLY_EXCEPTIONS** Whether the Q Apply inserted a row in the IBMQREP_EXCEPTIONS table because of a SQL error or conflict. **QAPPLY_MEMORY** Whether the amount of memory that the Q Apply program used to read messages from a particular receive queue exceeds a threshold. **QAPPLY_SPILLQDEPTH** Whether the number of messages on a spill queue exceeds a threshold. **QAPPLY_QDEPTH** Whether the number of messages on a receive queue exceeds a threshold.

Table 99. Columns in the IBMSNAP_CONDITIONS table (continued)

Column name	Description
PARM_INT	**Data type**: INT; **Nullable**: Yes. The integer parameter for the condition. The value of this column depends on the value of the CONDITION_NAME column. **CAPTURE_LASTCOMMIT** Threshold in seconds. **CAPTURE_CLATENCY** Threshold in seconds. **CAPTURE_HLATENCY** Threshold in seconds. **CAPTURE_MEMORY** Threshold in megabytes. **APPLY_SUBSDELAY** Threshold in seconds. **APPLY_REWORKED** Threshold in rows reworked. **APPLY_LATENCY** Threshold in seconds. **QCAPTURE_LATENCY** Threshold in seconds **QCAPTURE_MEMORY** Threshold in megabytes **QCAPTURE_TRANSIZE** Threshold in megabytes **QAPPLY_EELATENCY** Threshold in seconds **QAPPLY_LATENCY** Threshold in seconds **QAPPLY_MEMORY** Threshold in megabytes **QAPPLY_SPILLQDEPTH** Threshold in number of messages. **QAPPLY_QDEPTH** Threshold in number of messages.

Table 99. Columns in the IBMSNAP_CONDITIONS table (continued)

Column name	Description
PARM_CHAR	**Data type**: VARCHAR(128); **Nullable**: Yes. The character parameter for the condition. This column holds additional strings used by the condition. The CAPTURE_STATUS and APPLY_STATUS conditions use the value of this column. The value of this column is a string concatenating three parameters separated by commas: • Capture server or Apply control server. • Remote DB2 instance name (only when the server is remote). • Remote hostname. If the value is NULL or a zero length string, the Monitor program uses the following defaults: • The CURRENT SERVER value from the Capture or Apply control server. • The remote DB2 instance name value On UNIX servers, this value is the name of the user ID that was used when the server was connected. On Windows servers, the value is ″DB″. • The value of the hostname in the DB2 node directory.
CONTACT_TYPE	**Data type**: CHAR(1); **Nullable**: No. A flag that indicates whether to contact an individual or a group if this condition occurs: **C** Individual contact **G** Group of contacts
CONTACT	**Data type**: VARCHAR(127); **Nullable**: No. The name of the individual contact or the group of contacts to be notified if this condition occurs.

IBMSNAP_CONTACTGRP table

Server: Monitor control server

Index: GROUP_NAME, CONTACT_NAME

The IBMSNAP_CONTACTGRP table contains the individual contacts that make up contact groups. You can specify for the Replication Alert Monitor to contact these groups of individuals if an alert condition occurs. An individual can belong to multiple contact groups (the columns are not unique).

Table 100 provides a brief description of the columns in the IBMSNAP_CONTACTGRP table.

Table 100. Columns in the IBMSNAP_CONTACTGRP table

Column name	Description
GROUP_NAME	**Data type**: VARCHAR(127); **Nullable**: No. The name of the contact group.

Table 100. Columns in the IBMSNAP_CONTACTGRP table (continued)

Column name	Description
CONTACT_NAME	**Data type**: VARCHAR(127); **Nullable**: No. A contact name that is part of the group. These individuals are specified in the Monitor contacts (IBMSNAP_CONTACTS) table.

IBMSNAP_CONTACTS table

Server: Monitor control server

Index: CONTACT_NAME

The IBMSNAP_CONTACTS table contains the necessary information for the Replication Alert Monitor to use to notify individuals when an alert condition that is associated with the individuals (or their group) occurs. One individual per row is specified.

Table 101 provides a brief description of the columns in the IBMSNAP_CONTACTS table.

Table 101. Columns in the IBMSNAP_CONTACTS table

Column name	Description
CONTACT_NAME	**Data type**: VARCHAR(127); **Nullable**: No. The name of the contact. Only an individual contact is allowed. Group names are not supported.
EMAIL_ADDRESS	**Data type**: VARCHAR(128); **Nullable**: No. The main e-mail or pager address for this contact.
ADDRESS_TYPE	**Data type**: CHAR(1); **Nullable**: Yes. A flag that indicates whether the e-mail address for this contact is an e-mail account or a pager address: **E** The e-mail address is for an e-mail account. **P** The e-mail address is for a pager.
DELEGATE	**Data type**: VARCHAR(127); **Nullable**: Yes. The contact name to receive the notifications in a delegation period. Only an individual contact name is allowed. Group names are not supported.
DELEGATE_START	**Data type**: DATE; **Nullable**: Yes. The start date of a delegation period when notifications will be sent to the individual named in the DELEGATE column.
DELEGATE_END	**Data type**: DATE; **Nullable**: Yes. The end date of a delegation period.
DESCRIPTION	**Data type**: VARCHAR(1024); **Nullable**: Yes. A description of the contact.

IBMSNAP_MONENQ table

Server: Monitor control server

Index: MONITOR_QUAL

The IBMSNAP_MONENQ table is reserved for future options of DB2 replication.

Table 102 provides a brief description of the columns in the IBMSNAP_MONENQ table.

Table 102. Columns in the IBMSNAP_MONENQ table

Column name	Description
MONITOR_QUAL	**Data type**: CHAR(18); **Nullable**: No. Reserved for future options of DB2 replication.

IBMSNAP_GROUPS table

Server: Monitor control server

Index: GROUP_NAME

The IBMSNAP_GROUPS table contains the name and description of each contact group. One group per row is specified.

Table 103 provides a brief description of the columns in the IBMSNAP_GROUPS table.

Table 103. Columns in the IBMSNAP_GROUPS table

Column name	Description
GROUP_NAME	**Data type**: VARCHAR(127); **Nullable**: Yes. The name of the contact group.
DESCRIPTION	**Data type**: VARCHAR(1024); **Nullable**: Yes. A description of the contact group.

IBMSNAP_MONPARMS table

Server: Monitor control server

Index: MONITOR_QUAL

Default schema: ASN

This table contains information that you can update by using SQL.

The IBMSNAP_MONPARMS table contains parameters that you can modify to control the operations of the Replication Alert Monitor. You can define these parameters to set values such as the length of time that the Monitor program retains data in the CD and UOW tables before pruning and the number of notification messages that the Monitor program will receive whenever an alert

condition is met. If you make changes to the parameters in this table, the Monitor program reads your modifications only during startup.

Table 104 provides a brief description of the columns in the IBMSNAP_MONPARMS table.

Table 104. Columns in the IBMSNAP_MONPARMS table

Column name	Description
MONITOR_QUAL	**Data type**: CHAR(18); **Nullable**: No. The Monitor qualifier matches the parameters to the Replication Alert Monitor program to which these parameters apply.
ALERT_PRUNE_LIMIT	**Data type**: INT; **Nullable**: No, with default; **Default:** 10080 minutes (7 days). A flag that indicates how old the data is before it will be pruned from the table.
AUTOPRUNE	**Data type**: CHAR(1); **Nullable**: No, with default; **Default:** Y. A flag that indicates whether the Monitor program automatically prunes rows that are no longer needed from the CD, UOW, signal, trace, and Monitor tables: **Y** Autopruning is on. **N** Autopruning is off.
EMAIL_SERVER	**Data type**: INT(128); **Nullable**: Yes. The address of the e-mail server using the SMTP protocol.
LOGREUSE	**Data type**: CHAR(1); **Nullable**: No, with default; **Default:** N. A flag that indicates whether the Monitor program overwrites the Monitor log file or appends to it. **Y** The Monitor program reuses the log file by first deleting it and then re-creating it when the Monitor program is restarted. **N** The Monitor program appends new information to the Monitor log file.
LOGSTDOUT	**Data type**: CHAR(1); **Nullable**: No, with default; **Default:** N. A flag that indicates where the Monitor program directs the log file messages: **Y** The Monitor program directs log file messages to both the standard out (STDOUT) and the log file. **N** The Monitor program directs most log file messages to the log file only. Initialization messages go to both the standard out (STDOUT) and the log file.
NOTIF_PER_ALERT	**Data type**: INT; **Nullable**: No, with default; **Default:** 3. The number of notification messages that will be sent when an alert condition is met.
NOTIF_MINUTES	**Data type**: INT; **Nullable**: No, with default; **Default:** 60. The number of minutes that you will receive notification messages when an alert condition is met.
MONITOR_ERRORS	**Data type**: VARCHAR(128); **Nullable**: Yes. Specifies the e-mail address where notification messages are sent whenever an error occurs that is related to the operation of the Replication Alert Monitor.

Table 104. Columns in the IBMSNAP_MONPARMS table (continued)

Column name	Description
MONITOR_INTERVAL	**Data type**: INT; **Nullable**: No, with default; **Default:** 300 (5 minutes). How often, in seconds, that the Replication Alert Monitor is run to monitor the alert conditions that were selected.
MONITOR_PATH	**Data type**: VARCHAR(1040); **Nullable**: Yes. The path where the output from the Monitor program is sent.
RUNONCE	**Data type**: CHAR(1); **Nullable**: No, with default; **Default:** N. A flag that indicates whether the Monitor program will check for the alert conditions that were selected: **Y** The Monitor program checks for any alert conditions. **N** The Monitor program does not check for any alert conditions. If RUNONCE is set to y, then the MONITOR_INTERVAL is ignored.
TERM	**Data type**: CHAR(1); **Nullable**: No, with default; **Default:** N. A flag that indicates whether the Monitor program terminates when DB2 terminates: **Y** The Monitor program terminates when DB2 terminates: **N** The Monitor program stays active and waits for DB2 to be restarted.
TRACE_LIMIT	**Data type**: INT; **Nullable**: No, with default; **Default:** 10080. The number of minutes that rows remain in the IBMSNAP_MONTRACE table before they are eligible for pruning. During the pruning process, the rows in the Monitor trace table are pruned if the number of minutes (current timestamp – the time a row was inserted in the IBMSNAP_MONTRACE table) exceeds the value of TRACE_LIMIT.

IBMSNAP_MONSERVERS table

Server: Monitor control server

Index: MONITOR_QUAL, SERVER_NAME

The IBMSNAP_MONSERVERS table contains information about the last time that the Replication Alert Monitor monitored a Capture control server, Apply control server, Q Capture server, or Q Apply server.

Table 105 provides a brief description of the columns in the IBMSNAP_MONSERVERS table.

Table 105. Columns in the IBMSNAP_MONSERVERS table

Column name	Description
MONITOR_QUAL	**Data type**: CHAR(18); **Nullable**: No. The Monitor qualifier that identifies which Replication Alert Monitor was monitoring the Capture control server, Apply control server, Q Capture server, or Q Apply server.

Table 105. Columns in the IBMSNAP_MONSERVERS table (continued)

Column name	Description
SERVER_NAME	**Data type**: CHAR(18); **Nullable**: No. The name of the Capture control server, Apply control server, Q Capture server, or Q Apply server that the Replication Alert Monitor was monitoring.
SERVER_ALIAS	**Data type**: CHAR(8); **Nullable**: Yes. The DB2 UDB alias of the Capture control server, Apply control server, Q Capture server, or Q Apply server that the Replication Alert Monitor was monitoring.
LAST_MONITOR_TIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The time (at the Capture control server, Apply control server, Q Capture server, or Q Apply server) that the Replication Alert Monitor program last connected to this server. This value is used as a lower bound value to fetch messages from the control tables and is the same value that START_MONITOR_TIME from the last successful monitor cycle.
START_MONITOR_TIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The time (at the Capture control server, Apply control server, Q Capture server, or Q Apply server) that the Replication Alert Monitor connected to the Capture control server, Apply control server, Q Capture server, or Q Apply server. This value is used as a upper bound value to fetch alert messages from the control tables.
END_MONITOR_TIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The time (at the Capture control server, Apply control server, Q Capture server, or Q Apply server) that the Replication Alert Monitor ended monitoring this server.
LASTRUN	**Data type**: TIMESTAMP; **Nullable**: No. The last time (at the Monitor control server) when the Replication Alert Monitor started to process the Capture control server, Apply control server, Q Capture server, or Q Apply server.
LASTSUCCESS	**Data type**: TIMESTAMP; **Nullable**: Yes. The value from the LASTRUN column of the last time (at the Monitor control server) when the Replication Alert Monitor successfully completed processing the Capture control server, Apply control server, Q Capture server, or Q Apply server. If the monitoring of this server keeps failing, the value could be the same (the history of this columns is stored in the IBMSNAP_MONTRAIL table).
STATUS	**Data type**: SMALLINT; **Nullable**: No. A flag that indicates the status of the monitoring cycle: **-1** The Replication Alert Monitor failed to process this server successfully. **0** The Replication Alert Monitor processed this server successfully. **1** The Replication Alert Monitor is currently processing this server.

IBMSNAP_MONTRACE table

Server: Monitor control server

Index: MONITOR_QUAL, TRACE_TIME

The IBMSNAP_MONTRACE table contains audit trail information for the Replication Alert Monitor. Everything that the Monitor program does is recorded in this table, which makes it one of the best places for you to look if a problem with the Monitor program occurs.

Table 106 provides a brief description of the columns in the IBMSNAP_MONTRACE table.

Table 106. Columns in the IBMSNAP_MONTRACE table

Column name	Description
MONITOR_QUAL	**Data type**: CHAR(18); **Nullable**: No. The Monitor qualifier that identifies which Replication Alert Monitor issued the message.
TRACE_TIME	**Data type**: TIMESTAMP; **Nullable**: No. The timestamp when the message was inserted into this table.
OPERATION	**Data type**: CHAR(8); **Nullable**: No. A value used to classify messages: **ERROR** An error message **WARNING** A warning message **INFO** An informational message
DESCRIPTION	**Data type**: VARCHAR(1024); **Nullable**: No. The message code and text.

IBMSNAP_MONTRAIL table

Server: Monitor control server

Index: None

The IBMSNAP_MONTRAIL table contains information about each monitor cycle. The Replication Alert Monitor inserts one row for each Capture control server, Apply control server, Q Capture server, and Q Apply server that it monitors.

Table 107 provides a brief description of the columns in the IBMSNAP_MONTRAIL table.

Table 107. Columns in the IBMSNAP_MONTRAIL table

Column name	Description
MONITOR_QUAL	**Data type**: CHAR(18); **Nullable**: No. The Monitor qualifier that identifies which Replication Alert Monitor was monitoring the Capture control server, Apply control server, Q Capture server, or Q Apply server
SERVER_NAME	**Data type**: CHAR(18); **Nullable**: No. The name of the Capture control server, Apply control server, Q Capture server, or Q Apply server that the Replication Alert Monitor was monitoring.

Table 107. Columns in the IBMSNAP_MONTRAIL table (continued)

Column name	Description
SERVER_ALIAS	**Data type**: CHAR(8); **Nullable**: Yes. The DB2 UDB alias of the Capture control server, Apply control server, Q Capture server, or Q Apply server that the Replication Alert Monitor was monitoring.
STATUS	**Data type**: SMALLINT; **Nullable**: No. A flag that indicates the status of the monitoring cycle: **-1** The Replication Alert Monitor failed to process this server successfully. **0** The Replication Alert Monitor processed this server successfully. **1** The Replication Alert Monitor is currently processing this server.
LASTRUN	**Data type**: TIMESTAMP; **Nullable**: No. The time (at the Monitor control server) when the Replication Alert Monitor program last started to process the Capture control server, Apply control server, Q Capture server, or Q Apply server.
LASTSUCCESS	**Data type**: TIMESTAMP; **Nullable**: Yes. The last time (at the Monitor control server) when the Replication Alert Monitor successfully completed processing the Capture control server, Apply control server, Q Capture server, or Q Apply server.
ENDTIME	**Data type**: TIMESTAMP; **Nullable**: No, with default; **Default**: current timestamp. The time when this row was inserted into this table.
LAST_MONITOR_TIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The time (at the Capture control server, Apply control server, Q Capture server, or Q Apply server) when the Replication Alert Monitor last connected to the Capture control server, Apply control server, Q Capture server, or Q Apply server. This value is used as a lower bound value to fetch messages from the control tables and is the same value as START_MONITOR_TIME from the previous successful monitor cycle.
START_MONITOR_TIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The last time when the Replication Alert Monitor started to monitor the Capture control server, Apply control server, Q Capture server, or Q Apply server.
END_MONITOR_TIME	**Data type**: TIMESTAMP; **Nullable**: Yes. The last time when the Replication Alert Monitor finished monitoring the Capture control server, Apply control server, Q Capture server, or Q Apply server.
SQLCODE	**Data type**: INT; **Nullable**: Yes. The SQLCODE of any errors that occurred during this monitor cycle.
SQLSTATE	**Data type**: CHAR(5); **Nullable**: Yes. The SQLSTATE of any errors that occurred during this monitor cycle.
NUM_ALERTS	**Data type**: INT; **Nullable**: No. The number of alert conditions that occurred during this monitor cycle.
NUM_NOTIFICATIONS	**Data type**: INT; **Nullable**: No. The number of notifications that were sent during this monitor cycle.

Tables at the target server and their column descriptions

This section provides greater detail of each table used by the target server. It also lists and briefly describes the columns in each table. The table names are listed in alphabetical order, and the column names are listed in the order that they appear in each table from left to right.

Base aggregate table

schema.base_aggregate

Server: target server

Important: If you use SQL to update this table, you run the risk of losing your updates if a full refresh is performed by the Apply program.

A base aggregate table is a target table that contains the results of aggregate functions that are performed on data located at the source table.

Table 108 provides a brief description of the columns in the base aggregate table.

Table 108. Columns in the base aggregate table

Column name	Description
user columns	The aggregate data that was computed from the source table.
IBMSNAP_LLOGMARKER	The current timestamp at the source server when the aggregation of the data in the source table began.
IBMSNAP_HLOGMARKER	The current timestamp at the source server when the aggregation of the data in the source table completed.

Change aggregate table

schema.change_aggregate

Server: target server

Important: If you use SQL to update this table, you run the risk of losing your updates if a full refresh is performed by the Apply program.

A change aggregate table is a target table that contains the results of aggregate functions that are performed on data in the change-data (CD) table. This table is similar to the base aggregate table, except that the functions being performed at the CD table are done only for changes that occur during a specific time interval.

Table 109 provides a brief description of the columns in the change aggregate table.

Table 109. Columns in the change aggregate table

Column name	Description
user key columns	The columns that make up the target key.
user nonkey columns	The nonkey data columns from the source table. The column names in this target table do not need to match the column names in the source table, but the data types must match.
user computed columns	User-defined columns that are derived from SQL expressions. You can use computed columns with SQL functions to convert source data types to different target data types.

Table 109. Columns in the change aggregate table (continued)

Column name	Description
IBMSNAP_LLOGMARKER	The oldest IBMSNAP_LOGMARKER or IBMSNAP_LLOGMARKER value present in the (CD+UOW) or CCD table rows being aggregated.
IBMSNAP_HLOGMARKER	The newest IBMSNAP_LOGMARKER or IBMSNAP_HLOGMARKER value present in the (CD+UOW) or CCD table rows being aggregated.

Consistent-change data (CCD) table

schema.CCD_table

This table contains information that you can update by using SQL.

Server: target server

Important: If you use SQL to update this table, you run the risk of losing your updates if a full refresh is performed by the Apply program.

Consistent-change-data (CCD) tables are targets in subscription-set members that contain information about changes that occur at the source, and have additional columns to identify the sequential ordering of those changes. The values in the columns come from a join of the CD and UOW tables. A CCD table that is at the target server can be:

- An internal CCD table that acts as an alternative to the CD table.

  For change-capture replication, the Apply program applies changes to the targets directly from this table. The name of this type of CCD table is stored in the same row in the register (IBMSNAP_REGISTER) table as the replication source that it holds changes from.
- An external CCD that is a read-only target table.

  This type of CCD contains a audit trail of your source data at the target server.
- An external CCD that is a middle tier in a multi-tier replication configuration.

  This type of CCD is a target table for tier 1 and a source table for tier 3. The name of this type of CCD table is stored in its own row in the register (IBMSNAP_REGISTER) table.

For more information on the uses for CCD tables as targets, see "Selecting a target type" on page 69.

The Capture program does not insert data into CCD tables and does not prune them. Instead, your application requirements should determine the history retention period for CCD tables. Therefore, pruning of CCD tables is not automatic by default, but can be easily automated using an SQL statement to be processed after the subscription cycle.

For external CCDs, you can choose to include several columns from the UOW table: APPLY_QUAL, IBMSNAP_AUTHID, IBMSNAP_AUTHTKN, IBMSNAP_REJ_CODE, and IBMSNAP_UOWID.

The originally captured operation code in the IBMSNAP_OPERATION column and the sequence numbers IBMSNAP_INTENTSEQ and IBMSNAP_COMMITSEQ are included in CCD tables. For condensed CCD tables, only the latest values are kept for each row.

For information on CCD tables that are populated by Capture triggers or that contain non-relational data, see "*schema.CCD_table* (non-DB2)" on page 441.

Table 110 provides a brief description of the columns in the CCD table.

Table 110. Columns in the CCD table

Column name	Description
IBMSNAP_INTENTSEQ	The log or journal record sequence number that uniquely identifies a change. This value is globally ascending.
IBMSNAP_OPERATION	A flag indicating the type of operation for a record. **I** Insert **U** Update **D** Delete
IBMSNAP_COMMITSEQ	The log record sequence number of the captured commit statement. This value groups inserts, updates, and deletes by the original transactions for the source table.
IBMSNAP_LOGMARKER	The commit time at the Capture control server.
user key columns	If the CCD table is condensed, this column contains the columns that make up the target key.
user nonkey columns	The nonkey data columns from the source table. The column names in this target table do not need to match the column names in the source table, but the data types must be compatible.
user computed columns	User-defined columns that are derived from SQL expressions. You can use computed columns with SQL functions to convert source data types to different target data types.
IBMSNAP_APPLY_QUAL (optional)	Uniquely identifies which Apply program processes this CCD table.
IBMSNAP_AUTHID (optional)	The authorization ID associated with the transaction. It is useful for database auditing. For DB2 Universal Database for z/OS, this column is the primary authorization ID. For DB2 Universal Database for iSeries, this column has the name of the user profile ID under which the application that caused the transaction ran. This column holds a 10-character ID padded with blanks. This column is not automatically copied to other tables; you must select it and copy it as a user data column. This column can be selected as a user data column for a noncomplete CCD target table.
IBMSNAP_AUTHTKN (optional)	The authorization token associated with the transaction. This ID is useful for database auditing. For DB2 Universal Database for z/OS, this column is the correlation ID. For DB2 Universal Database for iSeries, this column is the job name of the job that caused a transaction. This column is not automatically copied to other tables; you must select it and copy it as a user data column. This column can be selected as a user data column for a noncomplete CCD target table.

Table 110. Columns in the CCD table (continued)

Column name	Description
IBMSNAP_REJ_CODE (optional)	This value is set only during update-anywhere replication if conflict detection is specified as standard or advanced when you define your replication source. It is not valid for non-DB2 relational targets because they cannot participate in update-anywhere configurations. **0** A transaction with no known conflict. **1** A transaction that contains a conflict where the same row in the source and replica tables have a change that was not replicated. When a conflict occurs, the transaction will be reversed at the replica table. **2** A cascade-rejection of a transaction dependent on a prior transaction having at least one same-row conflict. When a conflict occurs, the transaction will be reversed at the replica table. **3** A transaction that contains at least one referential-integrity constraint violation. Because this transaction violates the referential constraints defined on the source table, the Apply program will mark this subscription set as failed. Updates cannot be copied until the referential integrity definitions are corrected. **4** A cascade-rejection of a transaction dependent on a prior transaction having at least one constraint conflict.
IBMSNAP_UOWID (optional)	The unit-of-work identifier from the log record header for this unit of work.

Related reference:

- "*schema.CCD_table* (non-DB2)" on page 441

Point-in-time table

schema.point_in_time

Server: target server

Important: If you use SQL to update this table, you run the risk of losing your updates if a full refresh is performed by the Apply program.

The point-in-time table contains a copy of the source data, with an additional system column (IBMSNAP_LOGMARKER) containing the timestamp of approximately when the particular row was inserted or updated at the source server.

Table 111 provides a brief description of the columns in the point-in-time table.

Table 111. Columns in the point-in-time table

Column name	Description
user key columns	The columns that make up the target key.
user nonkey columns	The nonkey data columns from the source table or view. The column names in this target table do not need to match the column names in the source table, but the data types must match.
user computed columns	User-defined columns that are derived from SQL expressions. You can use computed columns with SQL functions to convert source data types to different target data types.
IBMSNAP_LOGMARKER	The approximate commit time at the Capture control server. This column is null following a full refresh.

Replica table

schema.replica

Server: target server

This table contains information that you can update by using SQL.

The replica table must have the same key columns as the source table. Because of these similarities, the replica table can be used as a source table for further subscription sets. Converting a target table into a source table is done automatically when you define a replica target type and specify the CHANGE DATA CAPTURE attribute. See "Defining read-write targets (update-anywhere)" on page 77 for more information.

Table 112 provides a brief description of the columns in the replica table.

Table 112. Columns in the replica table

Column name	Description
user key columns	The columns that make up the target key, which must be the same primary key as the master table.
user nonkey columns	The nonkey data columns from the source table. The column names in this target table do not need to match the column names in the source table, but the data types must match.

User copy table

schema.user_copy

Server: target server

Important: If you use SQL to update this table, you run the risk of losing your updates if a full refresh is performed by the Apply program.

The user copy table is a target table that contains a copy of the columns in the source table. This target table can be a row or column subset of the source table, but it cannot contain any additional columns.

Except for subsetting and data enhancement, a user copy table reflects a valid state of the source table, but not necessarily the most current state. References to user copy tables (or any other type of target table) reduce the risk of contention problems that results from a high volume of direct access to the source tables. Accessing local user copy tables is much faster than using the network to access remote source tables for each query.

Table 113 provides a brief description of the columns in the user copy table.

Table 113. Columns in the user copy table

Column name	Description
user key columns	The columns that make up the target key.
user nonkey columns	The nonkey data columns from the source table or view. The column names in this target table do not need to match the column names in the source table, but the data types must match.

Table 113. Columns in the user copy table (continued)

Column name	Description
user computed columns	User-defined columns that are derived from SQL expressions. You can use computed columns with SQL functions to convert source data types to different target data types.

Appendix A. UNICODE and ASCII encoding schemes for SQL replication (z/OS)

DB2 DataPropagator for OS/390 and z/OS Version 7 or later support both UNICODE and ASCII encoding schemes. To exploit the UNICODE encoding scheme, you must have at least DB2 for OS/390 and z/OS Version 7 and you must manually create or convert your DB2 DataPropagator source, target, and control tables as described in the following sections. However, your existing replication environment will work with DB2 DataPropagator for OS/390 and z/OS Version 7 or later even if you do not modify any encoding schemes. If your system is a UNICODE system, you must add ENCODING(EBCDIC) on the BIND PLAN and PACKAGE commands for the Capture, Apply, and Replication Alert Monitor programs. See "Planning for code page translation" on page 11 for more information on making code pages compatible and setting the code page variable.

Choosing an encoding scheme

If your source, CD, and target tables use the same encoding scheme, you can minimize the need for data conversions in your replication environment. When you choose encoding schemes for the tables, follow the single CCSID rule:

> The table space data is encoded using ASCII or EBCDIC or UNICODE CCSIDs. The encoding scheme of all the tables referenced by an SQL statement must be the same. Also, all tables that you use in views and joins must use the same encoding scheme.

If you do not follow the single CCSID rule, DB2 will detect the violation and return SQLCODE -873 during bind or execution.

Which tables should be ASCII or UNICODE depends on your client/server configuration. Specifically, follow these rules when you choose encoding schemes for the tables:

- Source or target tables on DB2 for OS/390 can be EBCDIC, ASCII, or UNICODE. They can be copied from or to tables that have the same or different encoding scheme in any supported DBMS (DB2 family, or non-DB2 with DataJoiner).
- On a DB2 for OS/390 source server, CD and UOW tables on the same server do not have to use the same encoding scheme if, when the subscription set-member is created, the target type is USERCOPY and JOIN_UOW_CD does not equal Y. Otherwise, the CD and UOW tables must use the same encoding scheme.
- The signal (IBMSNAP_SIGNAL) table should be encoded EBCDIC so that the Capture program does not have to translate signals to EBCDIC when it selects them from the signal table.
- All the control tables (ASN.IBMSNAP_SUBS_*xxxx*) on the same control server must use the same encoding scheme.
- Other control tables can use any encoding scheme.

Setting Encoding Schemes

To specify the proper encoding scheme for tables, modify the SQL that is used to generate the tables.

1. Create new source and target tables with the proper encoding scheme, or change the encoding schemes of the existing target and source tables. It is recommended that you stop the Capture and Apply programs before you change the encoding scheme of existing tables, and afterwards that you cold start the Capture program and restart the Apply program. To change the encoding scheme of existing tables (which must have the same encoding scheme within a table space):
   a. Use the Reorg Tablespace utility to unload the existing table space.
   b. Drop the existing table space.
   c. Re-create the table space specifying the new encoding scheme.
   d. Use the Load utility to load the old data into the new table space.

   See the *DB2 Universal Database for OS/390 and z/OS Utility Guide and Reference*, SC26-9945 for more information on the Load and Reorg utilities.
2. Use the Replication Center to create new control tables with the proper encoding scheme.
3. Use the Reorg and Load utilities to modify the encoding scheme for existing control tables and CD tables.
4. When you create new replication sources or subscription sets using the Replication Center, specify the proper encoding scheme.

The *SQL Reference*, SC26-9014 contains more information about CCSID.

Appendix B. How the Capture program processes journal entry types for SQL replication (iSeries)

The following table describes how the Capture program processes different journal entry types.

Table 114. Capture program processing by journal entry

Journal code[1]	Entry type	Description	Processing
C	CM	Set of record changes committed	Insert a record in the UOW stable.
C	RB	Rollback	No UOW row inserted.
F	AY	Journaled changes applied to physical file member	Issue an ASN2004 message and full refresh of file.
F	CE	Change end of data for physical file	Issue an ASN2004 message and full refresh of file.
F	CR	Physical file member cleared	Issue an ASN2004 message and full refresh of file.
F	EJ	Journaling for physical file member ended	Issue an ASN200A message and full refresh of the file. A full-refresh occurs whenever the Capture program reads an EJ journal entry, regardless of whether the user or the system caused journaling to end. Refer to the appropriate OS/400 documentation for information about implicit end-journal events for a file.
F	IZ	Physical file member initialized	Issue an ASN2004 message and full refresh of file.
F	MD	Member removed from physical file (DLTLIB, DLTF, or RMVM)	Issue an ASN200A message and attempt a full refresh.
F	MF	Storage for physical file member freed	Issue an ASN200A message and full refresh of file.
F	MM	Physical file containing member moved (Rename Object (RNMOBJ) of library, Move Object (MOVOBJ) of file)	Issue an ASN200A message and attempt a full refresh.
F	MN	Physical file containing member renamed (RNMOBJ of file, Rename Member (RNMM))	Issue an ASN200A message and attempt a full refresh.
F	MR	Physical file member restored	Issue an ASN2004 message and full refresh of file.
F	RC	Journaled changes removed from physical file member	Issue an ASN2004 message and full refresh of file.
F	RG	Physical file member reorganized	If the RRN of the source table is being used as the replication key, issue an ASN2004 message and full refresh of file.

Table 114. Capture program processing by journal entry (continued)

Journal code[1]	Entry type	Description	Processing
J	NR	Identifier for next journal receivers	Reset the Capture program.
J	PR	Identifier for previous journal receivers	Increment the unique sequence number counter.
R	DL	Record deleted from physical file member	Insert a DLT record in the CD table.
R	DR	Record deleted for rollback	Insert a DLT record in the CD table.
R	PT	Record added to physical file member	Insert an ADD record in the CD table.
R	PX	Record added directly to physical file member	Insert an ADD record in the CD table.
R	UB	Before-image of record updated in physical file member	See note 2.
R	UP	After-image of record updated in physical file member	See note 2.
R	BR	Before-image of record updated for rollback	See note 3.
R	UR	After-image of record updated for rollback	See note 3.

Notes:

1. The following values are used for the journal codes:
   - **C** Commitment control operation
   - **F** Database file operation
   - **J** Journal or journal receiver operation
   - **R** Operation on specific record
2. The R-UP image and the R-UB image form a single UPD record in the CD table if the PARTITION_KEYS_CHG column in the register table is N. Otherwise, the R-UB image inserts a DLT record in the CD table and the R-UP image inserts an ADD record in the CD table.
3. The R-UR image and the R-BR image form a single UPD record in the CD table if the PARTITION_KEYS_CHG column in the register table is N. Otherwise, the R-BR image inserts a DLT record in the CD table and the R-UR image inserts an ADD record in the CD table.

All other journal entry types are ignored by the Capture program.

Appendix C. Starting the SQL replication programs from within an application (Linux, UNIX, Windows)

You can start any of the replication programs (Capture program, Apply program, Replication Alert Monitor) for one replication cycle from within your application by calling routines. To use these routines you must specify the AUTOSTOP option for the Capture program and the COPYONCE option for the Apply program because the API support only synchronous execution.

Samples of the API and their respective makefiles are in the following directories:

For Windows
sqllib\samples\repl

For Linux and UNIX
sqllib/samples/repl

Those directories contain the following files for starting the Capture program:

capture_api.c
The sample code for starting the Capture program on Windows, Linux, or UNIX.

capture_api_nt.mak
The makefile for sample code on Windows.

capture_api_unix.mak
The makefile for sample code on UNIX.

Those directories contain the following files for starting the Apply program:

apply_api.c
The sample code for starting the Apply program on Windows, Linux, or UNIX.

apply_api_nt.mak
The makefile for sample code on Windows.

apply_api_unix.mak
The makefile for sample code on UNIX.

Those directories contain the following files for starting the Replication Alert Monitor:

monitor_api.c
The sample code for starting the Replication Alert Monitor on Windows, Linux, or UNIX.

monitor_api_nt.mak
The makefile for sample code on Windows.

monitor_api_unix.mak
The makefile for sample code on UNIX.

Glossary

Glossary

A

administration queue. In Q replication and event publishing, a WebSphere MQ queue that is used for communication between a Q Capture program and a Q Apply program or a user application. The administration queue for each Q Capture program must be a local, persistent queue.

after-image. In SQL replication, the updated content of a source-table column that is recorded in a change data (CD) table or in a database log or journal. Contrast with *before-image*.

after-value. In Q replication, the updated content of a source-table column.

agent thread. In Q replication, one of the threads of the Q Apply program that receives transactions from a browser thread and applies this data to target tables on the same server. There can be one or more agent threads for each browser thread.

aggregate table. In SQL replication, a read-only replication target table that contains aggregations of data from the source table. This data is based on SQL column functions such as MIN, MAX, SUM, or AVG.

alert. In replication, a notice that describes events and conditions in replication. The Replication Alert Monitor sends alerts via e-mail or pager.

alert condition. In replication, a condition of the replication environment that causes the Replication Alert Monitor to send alerts. There are three kinds of alert conditions: alert conditions that are triggered by status, alert conditions that are triggered by events, and alert conditions that are triggered by thresholds. You choose the alert conditions that the Replication Alert Monitor will check for in your replication environment.

apply. In replication, to refresh or update a replication target table.

Apply control server. In SQL replication, a database that contains Apply control tables that store information about subscription sets and subscription-set members. Contrast with *Apply server*.

Apply cycle. In SQL replication, the interval of time during which data is replicated from a source table to a target table.

Apply latency. In SQL replication, an approximate measurement of the time that replication requires to complete one cycle. See also *Capture latency*.

Apply program. In SQL replication, a program that is used to refresh or update a replication target table. Contrast with *Capture program* and *Capture trigger*.

Apply qualifier. In SQL replication, a case-sensitive character string that identifies replication subscription sets that are unique to an instance of the Apply program.

Apply server. In SQL replication, a system where the Apply program is running. Contrast with *Apply control server*.

archive log. (1) The set of log files that is closed and is no longer needed for normal processing. These files are retained for use in roll-forward recovery. (2) The portion of the DB2 Universal Database for z/OS log that contains log records that are copied from the active log. The archive log holds records that no longer fit on the active log.

ASP. See *auxiliary storage pool (ASP)*.

asynchronous replication. In replication, the process of copying data from a source table to a target table outside the scope of the original transaction that updated the source table. Contrast with *synchronous replication*.

audit trail. Data, in the form of a logical path that links a sequence of events. An audit trail traces the transactions that affect the contents of a record.

authorization token. (1) A token associated with a transaction. (2) For DB2 Universal Database for z/OS, the correlation ID. (3) For DB2 Universal Database for iSeries, the job name of the job that caused a transaction.

auxiliary storage pool (ASP). One or more storage units that are defined from the storage devices or storage device subsystems that make up auxiliary storage. An ASP provides a way of organizing data to limit the impact of storage-device failures and to reduce recovery time.

B

base aggregate table. In SQL replication, a type of replication target table that contains data that is aggregated from a replication source table. Contrast with *change aggregate table*.

before-image. In SQL replication, the content of a replication source-table column prior to an update by a transaction. The content is recorded in a change data (CD) table or in a database log or journal. Contrast with *after-image*.

before-value. In Q replication, the content of a replication source-table column prior to an update by a transaction.

bidirectional replication. In Q replication, a replication configuration in which changes that are made to one copy of a table are replicated to a second copy of that table. Changes that are made to the second copy are replicated back to the first copy. You must choose which copy of the table wins if a conflict occurs.

binary large object (BLOB). A data type that contains a sequence of bytes that can range in size from 0 bytes to 2 gigabytes less 1 byte. This string does not have an associated code page and character set. BLOBs can contain image, audio, and video data. See also *character large object* and *double-byte character large object*.

BLOB. See *binary large object (BLOB)*.

block fetch. A function of DB2 that retrieves (or fetches) a large set of rows together. Using block fetch can significantly reduce the number of messages that are sent across the network. Block fetch applies only to cursors that do not update data.

blocking. In SQL replication, an option that is specified when binding an application. It allows caching of multiple rows of information by the communications subsystem so that each FETCH statement does not require the transmission of one row for each request across the network. See also *block fetch*.

browser thread. In Q replication, a Q Apply program thread that gets messages from a receive queue and passes the messages on to one or more agent threads to be applied to targets.

C

capture. In replication, to gather changes from a source database for replication or event publishing.

Capture control server. (1) In SQL replication, a database that contains the Capture control tables, which store information about registered replication source tables. (2) A system where the Capture program is running.

Capture latency. In SQL replication, an approximate measurement of how recently the Capture program committed data to a CD table. See also *Apply latency*.

Capture program. In SQL replication, a program that reads database log or journal records to capture changes made to DB2 source tables. Contrast with *Apply program* and *Capture trigger*.

Capture schema. In SQL replication, a name that identifies the control tables used by a particular instance of the Capture program.

Capture trigger. In SQL replication, a mechanism that captures delete, insert, and update operations performed on non-DB2 relational source tables. Contrast with *Capture program* and *Apply program*.

cascade rejection. In SQL replication, the process of rejecting a replication transaction because it is associated with a transaction that had a conflict detected and was itself rejected.

CCD table. See *consistent-change data (CCD) table*.

CD table. See *change data table*.

change aggregate table. In SQL replication, a type of replication target table that contains data aggregations that are based on the contents of a CD table. Contrast with *base aggregate table*.

change-capture replication. In SQL replication, the process of capturing changes that are made to a replication source table and applying them to a replication target table. Contrast with *full refresh*.

change data (CD) table. In SQL replication, a replication table at the Capture control server that contains changed data for a replication source table.

character large object (CLOB). A data type that contains a sequence of characters (single-byte, multi-byte, or both) that can range in size from 0 bytes to 2 gigabytes less 1 byte. In general, character large object values are used whenever a character string might exceed the limits of the VARCHAR type. Also called character large object string. See also *binary large object* and *double-byte character large object (DBCLOB)*.

client. Any program or server where a program is running that communicates with and accesses a database server.

CLOB. See *character large object (CLOB)*.

cold start. (1) In SQL replication, the process of starting the Capture program without using restart information from prior operation of the program. When you perform a cold start, a full refresh occurs and all active subscriptions are deactivated and then activated. The cold start process deletes all transactions that were not processed before the cold start. Contrast with *warm start*. (2) In Q replication, the process of starting the Q Capture program without using restart information from prior operation of the program. When you perform a cold start a full refresh occurs. All transactions that were not processed before the cold start are processed after the cold start. The user is

responsible for cleaning transactions from the queues prior to the cold start. Contrast with *warm start*. (3) The process of starting a system or program by using an initial program load procedure. (4) A process by which DB2 Universal Database for z/OS restarts without processing any log records.

complete CCD table. In SQL replication, a CCD table that initially contains all the rows from the replication source table or view and any predicates from the source table or view. Contrast with *noncomplete CCD table*. See also *consistent-change data (CCD) table*.

condensed. In SQL replication, a table attribute that indicates that the table contains current data rather than a history of changes to the data. A condensed table includes no more than one row for each primary key value in the table. As a result, a condensed table can be used to supply current information for a refresh.

condensed CCD table. In SQL replication, a CCD table that contains only the most current value for a row and has only one row for each key value. Contrast with *noncondensed CCD table*. See also *consistent-change data (CCD) table*.

conflict detection. In bidirectional replication and update-anywhere replication, conflict detection refers to either of the following processes:

- The process of detecting constraint errors such as key constraints and referential constraints.
- The process of detecting whether the same row was updated by users or application programs in both the source and target tables during the same replication cycle.

consistent-change data (CCD) table. In SQL replication, a type of replication target table that is used for storing history, for auditing data, or for staging data. A CCD table can also be a replication source. See also *complete CCD table*, *condensed CCD table*, *external CCD table*, *internal CCD table*, *noncomplete CCD table*, and *noncondensed CCD table*.

Control Center. The DB2 graphical interface that lets you administrate DB2 databases and perform a variety of tasks including creating objects and monitoring performance. The Control Center shows database objects (such as databases and tables) and their relationship to each other.

control message. In Q replication, a message from a Q Apply program or a user application that asks a Q Capture program to activate or deactivate a Q subscription or an XML publication, invalidate a send queue, or confirm that a target table is loaded.

control server. In SQL replication, a database server that contains replication control tables for the Capture program, the Apply program, or the Replication Alert Monitor. See also *Apply control server*, *Capture control server*, *Q Apply server*, *Q Capture server*, and *Monitor control server*.

control table. See *replication control table*.

D

database management system (DBMS). See *database manager*.

database manager. A program that manages data by providing the services of centralized control, data independence, and complex physical structures for efficient access, integrity, recovery, data currency control, privacy, and security.

database recovery log. In replication, a set of primary and secondary log files that record all changes to a database in log records.

data blocking. In SQL replication, the process of replicating a specific number of minutes' worth of change data during an Apply cycle.

data distribution replication. In replication, a replication configuration that contains a single source table, from which changes are replicated to one or more read-only target tables. Before replication to the target tables can occur, the tables must contain a complete set of data from the source table.

data message. In Q replication, a message that contains one of the following things from the source table:

- All or part of a transaction.
- A single row operation.
- All or part of a large object (LOB) value from a row operation within a transaction.

DB2 replication. See *SQL replication*.

DBCLOB. See *double-byte character large object (DBCLOB)*.

DBMS. Database management system.

delimited identifier. A sequence of characters enclosed within double quotation marks ("). The sequence must consist of a letter followed by zero or more characters, each of which is a letter, a digit, or the underscore character. See also *ordinary identifier*.

differential-refresh replication. See *change-capture replication*.

distinct type. A user-defined data type that is internally represented as an existing type (its source type), but is considered to be a separate and incompatible type for semantic purposes. See also *user-defined type (UDT)*.

double-byte character large object (DBCLOB). A data type that contains a sequence of double-byte characters that can range in size from 0 bytes to 2 gigabytes. This data type can be used to store large double-byte text objects. Also called *double-byte character large object string*. Such a string always has an associated code page. See also *binary large object (BLOB)* and *character large object (CLOB)*.

E

end-to-end latency. In replication, an approximate measurement of the time that replication requires to capture changes from a source database and apply those changes to a target database. See also *Apply latency*, *Capture latency*, *Q Apply latency*, and *Q Capture latency*.

event publishing. A data publishing solution that captures transactional data from DB2 recovery logs and publishes that data as XML messages. The XML messages are published to WebSphere MQ queues where one or more user applications can retrieve and use those messages.

event timing. In SQL replication, the most precise method of controlling when to start a replication subscription cycle. Contrast with *interval timing*.

external CCD table. In SQL replication, a CCD table that can be subscribed to directly because it is a registered replication source. It has its own row in the register table, where it is identified by the SOURCE_OWNER and SOURCE_TABLE columns. See also *consistent-change data table*. Contrast with *internal CCD table*.

F

federated database system. A special type of distributed database management system (DBMS). A federated database system allows you to query and manipulate data located on other servers. The data can be in database managers such as Oracle, Sybase, Microsoft SQL Server, Informix, and Teradata or it can be in lists or stores such as a spreadsheet, Web site, or data mart. SQL statements can refer to multiple database managers or individual databases in a single statement. For example, you can join data located in a DB2 Universal Database table, an Oracle table, and a Sybase view.

full refresh. (1) In SQL replication, the process in which all of the data that matches the registration and the subscription-set predicates for a replication source table is copied to the target table. Also known as loading a target table. A full refresh replaces all existing data in the target table. Contrast with *change-capture replication*. (2) In Q replication, the process in which all of the data that matches the search conditions for a Q subscription for a replication source table is copied to the target table. A full refresh replaces all existing data in the target table.

G

gap. In SQL replication, a situation in which the Capture program is not able to read a range of log or journal records, so there is potential loss of change data.

global record. In SQL replication, the row in the register table that defines global replication characteristics for a particular instance of the Capture program.

H

heterogeneous replication. Replication between DB2 and non-DB2 relational databases. See also *federated database system*.

high-availability disaster recovery. A replication configuration where replicated data is available to dependent applications whenever required and protected against loss that is caused by catastrophic failure.

hot-spot update. A series of repeated updates made to the same rows in a short period of time.

I

IASP. See *Independent Auxiliary Storage Pool (IASP)*.

independent auxiliary storage pool (IASP). One or more storage units that are defined from the disk units or disk-unit subsystems that make up addressable disk storage. An independent auxiliary pool contains objects, the directories that contain the objects, and other object attributes such as authorization ownership attributes. An independent auxiliary storage pool can be made available (varied on) and made unavailable (varied off) without restarting the system. An independent auxiliary pool can either be switchable among multiple systems in a clustering environment or privately connected to a single system.

informational message. In Q replication and event publishing, a message that a Q Capture program sends to inform a Q Apply program or a user application about status of the Q Capture program, a Q subscription, or an XML publication.

internal CCD table. In SQL replication, a CCD table that cannot be subscribed to directly because it is not a registered replication source. It does not have its own row in the register table; it is identified by the CCD_OWNER and CCD_TABLE columns for the row

of the associated registered replication source. Contrast with *external CCD table*. See also *consistent-change data (CCD) table*.

interval timing. In SQL replication, the simplest method of controlling when to start a replication subscription cycle. When using interval timing, specify a date and a time for a subscription cycle to start, and set a time interval that describes how frequently the subscription cycle runs. Contrast with *event timing*.

J

join. An SQL relational operation that allows retrieval of data from two or more tables based on matching column values.

journal. For iSeries systems, a system object that identifies the objects being journaled, the current journal receiver, and all the journal receivers on the system for the journal. See also *journal receiver*.

journal code. For iSeries systems, a 1-character code in a journal entry that identifies the category of the journal entry. For example, `F` identifies an operation on a file, `R` identifies an operation on a record, and so forth. See also *journal entry*.

journal entry. For iSeries systems, a record in a journal receiver that contains information about a journaled change or other activity that is journaled. See also *journal code* and *journal entry type*.

journal entry type. For iSeries systems, a 2-character field in a journal entry that identifies the type of operation of a system-generated journal entry or the type of journal entry of a user-generated journal entry. For example, `PT` is the entry type for a write operation. See also *journal code*.

journal identifier (JID). For iSeries systems, a unique identifier that is assigned to a particular object when journaling is started for that object. Journal entries are associated with a particular object by this JID value.

journaling. For iSeries systems, the process of recording, in a journal, the changes made to objects, such as physical file members or access paths, or the depositing of journal entries by system or user functions.

journal receiver. For iSeries systems, a system object that contains journal entries added when events occur that are journaled, such as changes to a database file, changes to other journaled objects, or security-relevant events. See also *journal*.

K

key. (1) In replication, a column or an ordered collection of columns that is identified in the description of a table, index, or referential constraint. The same column can be part of more than one key. (2) In Q replication, the matching column or columns at both the source and target tables that are specified in the Q subscription.

L

large object (LOB). A data type that contains a sequence of bytes that can range in size from 0 bytes to 2 gigabytes less 1 byte. There are three types of large objects: binary large objects (binary), character large objects (single-byte character or mixed), and double-byte character large objects (double-byte character). See *binary large object (BLOB)*, *character large object (CLOB)*, and *double-byte character large object (DBCLOB)*.

latency. The time required for updates that were made to a source to replicate to a target.

load phase. In Q replication, the stage where a target table is loaded with data from a source table so that the two tables are synchronized. With an automatic load, the Q Apply program handles the loading process, and you can either specify a load utility or let the Q Apply program choose the best available utility. With a manual load, you load the target table and then notify the replication programs when the table is loaded.

LOB. See *large object (LOB)*.

logical server. In replication, on Linux, UNIX, and Windows, a DB2 database. On z/OS, a subsystem running DB2.

local database. A database that is physically located on the server in use. Contrast with *remote database*.

lock. (1) A means of serializing events or access to data. (2) A means of preventing uncommitted changes made by one application process from being perceived by another application process and for preventing one application process from updating data that is being accessed by another process.

locking. The mechanism that a database manager uses to ensure the integrity of data. Locking prevents concurrent users from accessing inconsistent data.

log. (1) A file used to record changes that are made in a system. (2) A collection of records that describe the events that occur during DB2 Universal Database for z/OS execution and that indicate their sequence. The information that is recorded is used for recovery in the event of a failure during DB2 Universal Database for z/OS execution. (3) See *database recovery log*.

M

master table. In SQL replication, specifically in update-anywhere replication, the original source table for data in the replica table. If replication conflict detection is enabled, changes made to the master table are retained, whereas changes made to the replica table are rejected. See also *update-anywhere replication*, *replica table*, and *conflict detection*.

member. See *subscription-set member*.

Monitor control server. In replication, a database that contains the Monitor control tables, which store information about alert conditions that the Replication Alert Monitor will monitor.

monitor qualifier. In replication, a case-sensitive character string that identifies a specific instance of the Replication Alert Monitor.

multidirectional replication. In Q replication, a replication configuration that includes peer-to-peer or bidirectional replication.

multi-tier replication. In SQL replication, a replication configuration in which changes are replicated from a replication source in one database to a replication target in another database, and changes from this replication target are replicated again to a replication target in another database.

N

nickname. (1) An identifier that a federated server uses to reference a data source object, such as a table or a view. (2) A name that is defined in a DB2 V8 for Informix sources database or DB2 II database to represent a physical database object (such as a table or stored procedure) in a non-DB2 database.

noncomplete CCD table. In SQL replication, a CCD table that is initially empty and has rows appended to it as changes are made to the replication source. Contrast with *complete CCD table*. See also *consistent-change data (CCD) table*.

noncondensed CCD table. In SQL replication, a CCD table that can contain more than one row for each key value. These duplicate rows represent the history of changes for the values in rows of a table. Contrast with *condensed CCD table*. See also *consistent-change data (CCD) table*.

non-DB2 relational database server. An Informix database server or a relational database server from a vendor other than IBM.

nullable. The condition in which a column, function parameter, or result can have an absence of a value.

null value. A parameter position for which no value is specified.

O

object. (1) Anything that can be created or manipulated with SQL—for example, tables, views, indexes, or packages. (2) In object-oriented design or programming, an abstraction that consists of data and operations associated with that data. (3) For NetWare, an entity that is defined on the network and thus given access to the file server. (4) In the Information Catalog Center, an item that represents a unit or distinct grouping of information. Each Information Catalog Center object identifies and describes information but does not contain the actual information. For example, an object can provide the name of a report, list its creation date, and describe its purpose.

occasionally connected. In SQL replication, a replication configuration that contains target servers that are not always connected to the network. This configuration allows users to connect to a primary data source for a short time to synchronize their local database with the data at the source.

ODBC. See *Open Database Connectivity (ODBC)*.

ODBC driver. A driver that implements ODBC function calls and interacts with a data source.

Open Database Connectivity (ODBC). An application program interface (API) that allows access to database management systems using callable SQL, which does not require the use of an SQL preprocessor. The ODBC architecture allows users to add modules, called *database drivers*, that link the application to their choice of database management systems at run time. Application programs do not need to be linked directly to the modules of all the supported database management systems.

ordinary identifier. (1) In SQL, a letter followed by zero or more characters, each of which is a letter (a-z and A-Z), a symbol, a number, or the underscore character, used to form a name. (2) In DB2 Universal Database for z/OS, an uppercase letter followed by zero or more characters, each of which is an uppercase letter, a number, a digit, or the underscore character.

P

package. A control structure produced during program preparation that is used to execute SQL statements.

peer-to-peer replication. In Q replication, a replication configuration between peer tables in which updates at any table are replicated to the other tables, and convergence is maintained. Peer-to-peer replication can

have two servers or three or more servers. Contrast with *update-anywhere replication*. See also *multi-tier replication*.

point-in-time table. In SQL replication, a type of replication target table whose content matches all or part of a source table, with an added column that identifies the approximate time when the particular row was inserted or updated at the source system.

predicate. An element of a search condition that expresses or implies a comparison operation.

primary key. A unique key that is part of the definition of a table. A primary key is the default parent key of a referential constraint definition. It is a column or combination of columns that uniquely identifies a row in a table.

promote. In SQL replication, to copy replication definitions for subscription sets or registered sources from one database to another database, without having to register the sources again or create the subscription sets again.

pruning. In replication, the task of removing obsolete data from replication control tables or log files that are used by the Capture, Q Capture, Apply, and Q Apply programs.

publishing queue map. In event publishing, an object that includes a send queue for sending messages and settings for how a Q Capture program processes all transactions that use the send queue. See also *replication queue map* and *queue map*

pull configuration. In SQL replication, a replication configuration in which the Apply program runs at the target server. The Apply program pulls updates from the source server to apply them to the target. Contrast with *push configuration*.

push configuration. In SQL replication, a replication configuration in which the Apply program runs at the source server or a replication server other than the target server. The Apply program pushes updates from the source server to apply them to the target. Contrast with *pull configuration*.

Q

Q Apply latency. In Q replication, the time it takes for a transaction to be applied to a target table after the Q Apply program gets the transaction from a receive queue.

Q Apply program. In Q replication, a program that reads transactions from a receive queue and applies those changes to one or more target tables or passes the changes to a stored procedure.

Q Apply schema. In Q replication, the identifier for a Q Apply program and its control tables.

Q Apply server. In Q replication, a database or subsystem on which the control tables for the Q Apply program are located and where the Q Apply program runs. It contains one or more sets of the control tables that store information about target tables and other replication definitions.

Q Capture latency. In Q replication, an approximate measure of how current a Q Capture program is in reading the DB2 recovery log. Q Capture latency measures the time between a Q Capture program saving performance data and the timestamp of the last committed transaction that the program had read in the log when it saved the data. For example, if the Q Capture program saved performance data at 10 a.m. and the timestamp of the last committed transaction was 9:59 a.m., the Q Capture latency would be one minute.

Q Capture program. In Q replication and event publishing, a program that reads the DB2 recovery log to capture changes made to DB2 source tables and transmits the changes via one or more send queues.

Q Capture schema. In Q replication, the identifier for a Q Capture program and its control tables.

Q Capture server. In Q replication and event publishing, a database or subsystem on which the control tables for the Q Capture program are located and where the Q Capture program runs. It contains one or more sets of the control tables that store information about Q subscriptions and XML publications and other replication or publishing definitions.

Q Capture transaction latency. In Q replication, the time from when a Q Capture program reads the commit statement for a transaction in the DB2 recovery log, to the time when the Q Capture program puts the message containing the transaction on a send queue.

Q replication. A replication solution that uses WebSphere MQ message queues for high-volume, low-latency replication. It provides a peer-to-peer solution with conflict detection, conflict resolution, and convergence.

Q subscription. In Q replication, an object that identifies a mapping between a source table and target table or stored procedure and specifies what changes are replicated. The Q Capture program replicates changes from a source table and puts those changes on a send queue in compact format. Then, the Q Apply program gets the compact messages from a receive queue and applies the changes to a target table or passes them to a stored procedure for data manipulation. Q subscriptions are separate objects from XML publications in that Q subscriptions do not replicate data that is published in an XML publication.

Q subscription group. In Q replication, the group of Q subscriptions that are involved in replicating the same logical tables.

queue. A WebSphere MQ object. Message queuing applications can put messages on, and get messages from, a queue. The Q Capture and Q Apply programs can put messages on, and get messages from a queue. A queue is owned and maintained by a queue manager.

queue latency. In Q replication and event publishing, the time between the Q Capture program putting a transaction on a send queue and the Q Apply program getting the transaction from the receive queue.

queue map. In Q replication and event publishing, an object that links queues and defines how the Q Capture and Q Apply programs process messages that use the queues. There are two kinds of queue maps: publishing queue maps and replication queue maps. See also *publishing queue map* and *replication queue map*.

R

RDBMS. See *relational database management system (RDBMS)*.

real-time replication. See *synchronous replication*.

receive queue. In Q replication, a WebSphere MQ message queue that is used by a Q Apply program to receive transactions that are captured by a Q Capture program.

recapture. In update-anywhere replication, to capture changes at a replica table and forward these changes to the master table or to other replica tables.

referential constraints. The referential integrity rule that the non-null values of the foreign key are valid only if they also appear as values of a parent key.

referential integrity. The state of a database in which all values of all foreign keys are valid. Maintaining referential integrity requires the enforcement of *referential constraints* on all operations that change the data in a table where the referential constraints are defined.

register. In SQL replication, to define a DB2 table, view, or nickname as a replication source.

registration. (1) In SQL replication, the process of registering a DB2 table, view, or nickname as a replication source. Contrast with *subscription*. (2) See *replication source*.

rejected transaction. A transaction containing one or more updates from replica tables that are in conflict with the master table.

relational database management system (RDBMS) . A collection of hardware and software that organizes and provides access to a relational database.

remote database. A database that is physically located on a server other than the one in use. Contrast with *local database*.

replica table. In SQL replication, specifically in update-anywhere replication, a type of target table that can be updated locally and also receives updates from the master table through a subscription-set definition. If replication conflict detection is enabled, changes made to the replica table are rejected, whereas changes made to the master table are retained. See also *update-anywhere replication*, *master table*, and *conflict detection*.

replication. The process of maintaining a defined set of data in more than one location. It involves copying designated changes for one location (a source) to another (a target), and synchronizing the data in both locations.

replication administrator. (1) In Q replication, the user responsible for creating Q subscriptions and XML publications. This user can also run the Q Capture program and the Q Apply program (2) In SQL replication, the user responsible for registering replication sources and creating subscription sets. This user can also run the Capture program and the Apply program.

Replication Alert Monitor. In replication, a program that checks the operation of the Capture, Apply, Q Capture, and Q Apply programs, and sends alerts to one or more users when it detects the specified alert conditions.

Replication Analyzer. In replication, a program that can analyze a replication environment for setup problems, configuration errors, and performance issues.

Replication Center. In replication, a graphical user interface that lets you define, operate, maintain and monitor the replication environment. It is part of the DB2 Administration Client tool suite.

replication control table. In replication, a table in which replication definitions or control information is stored.

replication queue map. In Q replication, an object that links a send queue and a receive queue. The replication queue map includes settings for how a Q Capture program processes all transactions that use the send queue and how a Q Apply program processes all transactions that use the receive queue. See also *publishing queue map* and *queue map*.

replication source. (1) In SQL replication, a table, view, or nickname that is registered as a source for replication. Changes that are made to this table are

captured and copied to a target table that is defined in a subscription-set member. See also *subscription set* and *subscription-set member*. (2) In Q replication, a table that is a source for replication. Changes made to this type of table are captured and copied to a target table that is defined in a Q subscription or an XML publication. See also *Q subscription* and *XML publication*.

replication target. (1) In SQL replication, a table, view, or nickname that is a destination for changes that were replicated from a registered replication source. The Apply program applies these changes. See also *target table*. (2) In Q replication, a table or stored procedure that is a destination for changes that were replicated from a source. The Q Apply program applies these changes. See also *target table*.

retention-limit pruning. In SQL replication, the pruning of CD and UOW tables by the Capture program that are older than a limit that the user specifies.

rework. (1) To convert an insert into a replication target table to an update if the insert fails because the row already exists in the target table. (2) To convert an update to a replication target table to an insert if the update fails because the row does not exist in the target table.

row-capture rules. In SQL replication, rules based on changes to registered columns that define when and whether the Capture program writes a row to a CD table, or when and whether the Capture triggers write a row to a CCD table.

S

send queue. In Q replication, a WebSphere MQ message queue that is used by a Q Capture program to publish transactions that it has captured. A send queue can be used either for Q replication or event publishing, but not both at the same time.

serialization. (1) The consecutive ordering of items. (2) The process of controlling access to a resource to protect the integrity of the resource. (3) In Q replication, the process of applying transactions in the same order that they were committed at the source.

server. See *logical server* See also: *Apply control server*, *Apply server*, *Capture control server*, *Control server*, *Monitor control server*, *Q Apply server*, *Q Capture server*, *source server*, and *target server*.

signal. A communication mechanism for replication that allows communication with the Capture program and the Q Capture program. A signal is an SQL statement that is inserted into the signal control table, and received by the Capture program or the Q Capture program when it reads the log entry for the signal insert.

source server. In replication, a database or subsystem that contains the source tables.

source table. In replication, a table that contains data that is to be replicated to a target table. Contrast with *target table*.

spill agent thread. In Q replication, a thread that applies transactions that are waiting in the spill queue and informs the browser thread once the spill queue is empty and has been deleted.

spill file. In SQL replication, a temporary file created by the Apply program that is used to hold data for updating target tables.

spill queue. In Q replication, a dynamic queue that the Q Apply program creates to hold transactions that occur at the source table while a target table is being loaded. The Q Apply program later applies these transactions and then deletes the spill queue.

SQL replication. A type of replication that uses staging tables.

staging table. In SQL replication, a CCD table that is used to save data before that data is replicated to the target database. A CCD table that is used for staging data can function as an intermediate source for updating data to one or more target tables. See also *consistent-change-data table*.

subscription. (1) In SQL replication, an object that creates subscription sets and subscription-set members. Contrast with *registration* in SQL replication, and *Q subscription* in Q replication. (2) See also *subscription set*.

subscription cycle. The process in which SQL replication retrieves changed data for a given subscription set, replicates the changes to the target table, and updates the appropriate replication control tables to reflect its status and current progress.

subscription set. In SQL replication, a replication definition that controls the replication of changed data during a subscription cycle. A subscription set can contain zero or more subscription-set members.

subscription-set member. In SQL replication, a replication definition that maps a registered replication source with a replication target. Each member defines the structure of the target table and the rows and columns that will be replicated from the source table.

subset. To replicate data from part of a source table, rather than from the entire table, to a target table. You can subset by rows or by columns.

synchpoint. In SQL replication, a replication control table value for the DB2 log or journal record sequence number of the last change applied during the most recent Apply cycle. This value is also used to coordinate the pruning of CD tables.

synchronous replication. Also known as real-time replication, a type of replication that delivers updates continuously and within the scope of source transactions.

T

table-mode processing. In SQL replication, a type of replication subscription-set processing in which the Apply program retrieves all the data from the source CD table, then applies the data (one member at a time) to each target table, and finally commits its work. Contrast with *transaction-mode processing*.

target server. (1) In SQL replication, a database or subsystem that contains replication target tables, views, or stored procedures. (2) In Q replication, a database or subsystem that contains replication target tables or stored procedures.

target table. (1) In SQL replication, a table that is the destination for replicated changes from a registered replication source. It can be a user copy table, a point-in-time table, a base aggregate table, a change aggregate table, a CCD table, or a replica table. (2) In Q replication, a table that is the destination for replicated changes from a source that is part of a Q subscription.

timestamp. A data type, that contains a seven-part value that consists of a date and time expressed in years, months, days, hours, minutes, seconds, and microseconds.

trace. (1) For replication, a facility that provides the ability to collect monitoring, auditing, and performance data for the Capture program, the Q Capture program, Apply program, the Q Apply program, or Replication Alert Monitor. (2) A DB2 Universal Database for z/OS facility that provides the ability to monitor and collect monitoring, auditing, performance, accounting, statistics, and serviceability (global) data.

transaction. An exchange between a server and a program, two servers, or two programs that accomplishes a particular action or result. An example of a transaction is the entry of a customer's deposit and the subsequent update of the customer's balance. Synonym for *unit of work*.

transaction-based replication. In SQL replication, a type of replication processing in which every transaction is replicated to the target table when it is committed in the source table. Contrast with *transaction-consistent replication*.

transaction-consistent replication. In SQL replication, a type of replication processing in which the net result of all transaction updates is replicated to the target table. Contrast with *transaction-based replication*.

transaction-mode processing. In SQL replication, a type of replication subscription-set processing in which the Apply program retrieves data from the source CD table, then applies the data to the target table in the same commit sequence used at the source. The Apply program processes transactions for all subscription-set members together, rather than sequentially. Contrast with *table-mode processing*.

trigger. (1) An object in a database that is invoked indirectly by the database manager when a particular SQL statement is run. See also *Capture trigger*. (2) A set of SQL statements that is stored in a DB2 database and executed when a certain event occurs in a DB2 table.

U

UDT. See *user-defined type (UDT)*.

uncommitted read (UR). An isolation level that allows an application to access uncommitted changes of other transactions. The application does not lock other applications out of the row that it is reading unless the other application attempts to drop or alter the table.

Unicode. An international character encoding scheme that is a subset of the ISO 10646 standard. Each character supported is defined using a unique 2-byte code.

unidirectional replication. In Q replication, a replication configuration in which changes that occur at a source table are replicated over WebSphere MQ queues to a target table or are passed to a stored procedure to manipulate the data. Changes that occur at the target table are not replicated back to the source table.

unique index. An index that ensures that no identical key values are stored in a table.

unique key. A key that is constrained so that no two of its values are equal.

unit of work. (1) A recoverable sequence of operations within an application process. At any time, an application process is a single unit of work, but the life of an application process can involve many units of work as a result of commit or rollback operations. In a DB2 Universal Database for z/OS multisite update operation, a single unit of work can include several units of recovery. Synonym for *transaction*. (2) In the Information Catalog Center, a recoverable sequence of operations within an application process. At any time, an application process is a single unit of work, but the life of an application process can involve many units of work as a result of commit or rollback operations.

unit-of-work (UOW) table. In SQL replication, a replication control table stored in the Capture control server that contains commit records read from the database log or journal. The records show that a transaction or UOW committed successfully and include a unit-of-recovery ID that can be used to join

the unit-of-work table and the CD table to produce transaction-consistent change data.

update-anywhere replication. In SQL replication, a replication configuration in which all tables are both registered sources and read-write targets. One table is the primary source table for full refresh of all the others. In this configuration, there is an implicit replication hierarchy among the source and target tables. Contrast with *peer-to-peer replication*. See also *multi-tier replication*, *master table*, and *replica table*.

user copy table. In SQL replication, a replication target table whose content matches all or part of a registered source table and contains only user data columns.

user-defined type (UDT). A data type that is not native to the database manager and was created by a user. In DB2 Universal Database, the term *distinct type* is used instead of user-defined type.

V

view. (1) A logical table that consists of data that is generated by a query. A view is based on an underlying set of base tables, and the data in a view is determined by a select type query that is run on the base tables. (2) A way of looking at the information about, or contained in objects. Each view might reveal different information about its objects.

W

warm start. In replication, the process of starting the Capture program or the Q Capture program so that it reads transactions from the point where it left off. Contrast with *cold start*.

work file. In SQL replication, a temporary file used by the Apply program when processing a subscription set.

X

XML publication. In event publishing, an object that identifies what changes are published from a source table to a user application. The Q Capture program publishes changes from a source table and puts those changes on a send queue in XML format. You provide an application other than the Q Apply program to retrieve and use those XML messages. XML publications are separate objects from Q subscriptions in that Q subscriptions do not replicate data that is published in an XML publication.

Index

Special characters

; delimiter 102
$TA JES2 command 413
*.APP.log file 127
*.CAP.log file 111
*.err file 130
*.sqs file 129, 130
delimiter 102

A

abstract data types 85
activating subscription sets 61, 235
add_partition parameter
  overview 108
  use with asncap command 283
ADDDPRREG command 319
ADDDPRSUB command 327
ADDDPRSUBM command 343
ADDEXITPGM command 34
ADDJOBSCDE command 414
administration
  authorization requirements 17
after-image columns 41
aggregate tables
  base aggregate 72, 501
  change aggregate 73, 501
alert conditions
  ASNMAIL exit routine 150
  e-mail notification 149
  for Apply program 146
  for Capture program 146
  for Q Apply program 146
  for Q Capture program 146
  list 146
  notification criteria 162
  overview 145
  selecting 155
Alert Monitor
  See Replication Alert Monitor
alert_prune_limit parameter, Replication Alert Monitor 158
ALWINACT parameter 390
Analyzer
  for OS/400
    creating SQL packages 29
    invocation parameters 353
  for UNIX, invocation parameters 273
  for Windows, invocation parameters 273
Analyzer report
  ANZDPR command 352
  asnanalyze command 273
ANZDPR command 352
ANZDPRJRN command 33
APPENQ (Apply enqueue) table 463
APPLHEAPSZ configuration parameter 26
applications
  starting replication programs from 511
Apply control server
  adding to Replication Center 229
  control tables at 463
Apply control tables
  APPENQ (Apply enqueue) 463
  APPLY_JOB (Apply job) 464
  APPLYTRACE (Apply trace) 467
  APPLYTRAIL (Apply trail) 468
  APPPARMS (Apply parameters) 465
    changing 132
    using 122
  list of 463
  SUBS_COLS (subscription columns) 473
  SUBS_EVENT (subscription events) 475
  SUBS_MEMBR (subscription members) 475
  SUBS_SET (subscription sets) 480
  SUBS_STMTS (subscription statements) 485
Apply enqueue (APPENQ) table 463
Apply job (APPLY_JOB) table 464
Apply parameters (APPPARMS) table 465
  changing 132
  using 122
Apply program
  alert conditions 146
  authorization requirements 19
  changing parameter values 122
  commands 271
  communicating with
    Capture program 415, 416
    Capture triggers 415, 417
    Replication Alert Monitor 419
    Replication Center 415
  connectivity 15
  data blocking 61
  for OS/400
    ALWINACT parameter 390
    APYQUAL parameter 388
    checking status 166
    COPYONCE parameter 390
    creating SQL packages 29
    CTLSVR parameter 388
    DELAY parameter 390
    FULLREFPGM parameter 389
    INACTMSG parameter 389
    JOBD parameter 387
    OPTSNGSET parameter 391
    RTYWAIT parameter 390
    scheduling 414
    setting up 28, 30
    starting 131, 386
    stopping 133, 361
    SUBNFYPGM parameter 389
    TRACE parameter 388
Apply program *(continued)*
  for OS/400 *(continued)*
    TRLREUSE parameter 391
    USER parameter 387
  for UNIX
    apply_path parameter 124, 277
    apply_qual parameter 125, 272, 277
    binding 27
    checking status 165
    configuring 26
    control_server parameter 125, 272, 277
    copyonce parameter 125, 279
    default parameters 121
    delay parameter 126, 280
    errwait parameter 126, 280
    inamsg parameter 127, 278
    loadxit parameter 127, 278
    logreuse parameter 127, 278
    logstdout parameter 127, 278
    notify parameter 128, 278
    operating 271
    opt4one parameter 128, 280
    password file 22
    pwdfile parameter 128, 277
    setting up 25
    sleep parameter 128, 279
    spillfile parameter 129, 281
    sqlerrcontinue parameter 129, 281
    starting 123, 276, 511
    status 271
    stopping 133, 271
    term parameter 130, 280
    trlreuse parameter 130, 279
  for Windows
    apply_path parameter 124, 277
    apply_qual parameter 125, 272, 277
    binding 27
    checking status 165
    configuring 26
    control_server parameter 125, 272, 277
    copyonce parameter 125, 279
    default parameters 121
    delay parameter 126, 280
    errwait parameter 126, 280
    inamsg parameter 127, 278
    loadxit parameter 127, 278
    logreuse parameter 127, 278
    logstdout parameter 127, 278
    notify parameter 128, 278
    operating 121, 271
    opt4one parameter 128, 280
    password file 22
    pwdfile parameter 128, 277
    setting up 25
    sleep parameter 128, 279
    spillfile parameter 129, 281
    sqlerrcontinue parameter 129, 281

Apply program *(continued)*
for Windows *(continued)*
starting 123, 276, 511
status 271
stopping 133, 271
term parameter 130, 280
trlreuse parameter 130, 279
for z/OS
apply_path parameter 124, 277
apply_qual parameter 125, 272, 277
checking status 165
control_server parameter 125, 272, 277
copyonce parameter 125, 279
db2_subsystem parameter 126, 277
default parameters 121
delay parameter 126, 280
errwait parameter 126, 280
inamsg parameter 127, 278
loadxit parameter 127, 278
logreuse parameter 127, 278
logstdout parameter 127, 278
notify parameter 128, 278
operating 271
opt4one parameter 128, 280
pwdfile parameter 128, 277
setting up 30
sleep parameter 128, 279
spillfile parameter 129, 281
starting 123, 276
status 271
stopping 133, 271
term parameter 130, 280
trlreuse parameter 130, 279
latency analysis 170
messages 169
printing 169
mini-cycles 61
operating 239
performance data 167
run-time processing statements 98
scheduling 413
setting defaults for parameters 122
spill files, storage requirements 8
table-mode processing 64
throughput analysis 170
transaction-mode processing 64
user ID 19
Apply qualifiers
changing in subscription sets 192
monitoring status 170
naming rules 269
number of associated subscription sets 58
use when starting the Apply program 123, 131
Apply trace (APPLYTRACE) table
pruning 214
structure 467
Apply trail (APPLYTRAIL) table
pruning 214
structure 468
APPLY_JOB (Apply job) table 464
apply_path parameter 124, 277
apply_qual parameter 125, 272, 277
Apply-qualifier cross-reference (AUTHTKN) table 434
APPLYTRACE (Apply trace) table
pruning 214
structure 467
APPLYTRAIL (Apply trail) table
pruning 214
structure 468
APPPARMS (Apply parameters) table 465
changing 132
using 122
APYQUAL parameter 388
ASCII tables 507
asnacmd command 271
asnanalyze command 273
asnapply command 276
asncap command 282
asnccmd command 288
ASNDLCOPY exit routine 88
ASNDLCOPYD file-copy daemon 91
ASNDONE exit routine
rejected transactions 50
using 133, 134
asndone.smp file 134
ASNLOAD exit routine
customizing behavior 138
description 135
error handling 135
files generated 137
for DATALINK replication 87
for OS/400 140
for UNIX 136
for Windows 136
for z/OS 137
prerequisites 135
using asnload.ini file 139
using load from cursor function 139
asnload.ini file 139
ASNMAIL exit routine 150
asnslist command 306
asntdiff command 307
asntrep command 316
AT command
Apply program 413
Capture program 413
Replication Alert Monitor 413
AT NetView command
Apply for z/OS 413
Capture for z/OS 413
attributes
changing for registered objects 174
changing for subscription sets 183
auditing
cold start 73
gap in data 73
source data 42
authentication, end-user
for UNIX 15, 22
for Windows 15, 22
authorization
for administration 17, 18
for Apply program 19
for Capture program 18
for Capture triggers 19
for Replication Alert Monitor 21
AUTHTKN (Apply-qualifier cross-reference) table 434
automatic pruning 212
autoprune parameter
overview 108
use with asncap command 284
use with asnccmd command 289
autoprune parameter, Replication Alert Monitor 158
autostop parameter 109, 284, 289

B

backup database command 26
base aggregate tables
definition 69
structure 501
usage 72
batch jobs
memory used by 3
before-image columns
change-aggregate tables 81
registering 41
restrictions 42
before-image prefix 43
binary large object (BLOB)
replication considerations 85
binding
Apply program
for UNIX 27
for Windows 27
for z/OS 30
Capture program
for UNIX 26
for Windows 26
for z/OS 30
Replication Alert Monitor
for UNIX 27
for Windows 27
BLOB (binary large object)
replication considerations 85
blocking factor 61

C

calculated columns 80
CALL procedures
before and after run-time processing 98
defining for subscription set 65
CAPCTLLIB parameter 396
CAPENQ (Capture enqueue) table 435
CAPMON (Capture monitor) table
pruning 214
structure 436
CAPPARMS (Capture parameters) table
changing 117
structure 437
using 105
CAPSCHEMAS (Capture schemas) table 434
CAPSTART signals 201
CAPSTOP signals 202
CAPTRACE (Capture trace) table
pruning 214
structure 440

Capture
multiple database partitions 30
using multiple database partitions 24
Capture control server
adding to Replication Center 229
control tables at 433
multiple Capture schemas 24
Capture control tables
AUTHTKN (Apply-qualifier cross-reference) 434
CAPENQ (Capture enqueue) 435
CAPMON (Capture monitor) 436
CAPPARMS (Capture parameters)
changing 117
structure 437
using 105
CAPSCHEMAS (Capture schemas) 434
CAPTRACE (Capture trace) 440
CCD (consistent-change-data) 441
CD (change-data) 442
list of 433
PARTITIONINFO (partition information) 443
PRUNCNTL (pruning control) 444
PRUNE_LOCK (prune lock) 446
PRUNE_SET (prune set) 447
REG_EXT (register extension) 447
REG_SYNCH (register synchronization) 455
REGISTER (register) 449
RESTART (restart) 456
SEQTABLE (sequencing) 458
SIGNAL (signal) 458
UOW (unit-of-work) 461
Capture enqueue (CAPENQ) table 435
Capture log file 111
Capture monitor (CAPMON) table
pruning 214
structure 436
Capture parameters (CAPPARMS) table
changing 117
structure 437
using 105
Capture program
alert conditions 146
altering behavior while running 116
authorization requirements 18
changing parameter values 105
changing schemas 179
cold start prevention 215
commands 271
communicating with
Apply program 415, 416
Replication Alert Monitor 419
Replication Center 415
connectivity 15
for OS/400
authorization requirements 18
CAPCTLLIB parameter 396
changing attributes 355
checking status 166
CLNUPITV parameter 395
cold start parameters 394
cold start, automatic 399
creating SQL packages 28, 29
default parameters 104, 105
Capture program *(continued)*
for OS/400 *(continued)*
FRCFRQ parameter 398
JOBD parameter 394
journal entry types 509
journals and journal receivers, managing 32
JRN parameter 396
LAG parameter 398
MEMLMT parameter 397
MONITV parameter 397
MONLMT parameter 396
operating 103
overriding attributes of 375
progress of 171
reinitializing 374
RESTART parameter 394
RETAIN parameter 397
scheduling 414
setting up 28, 30
starting 115, 393
stopping 118, 364
TRCLMT parameter 396
WAIT parameter 395
warm start parameters 394
for UNIX
add_partition parameter 108, 283
autoprune parameter 108, 284, 289
autostop parameter 109, 284, 289
binding 26
capture_path parameter 109, 283
capture_schema parameter 110, 283
capture_server parameter 110, 283
changing parameters 288
checking status 165
cold start parameters 114, 286
commit_interval parameter 110, 284, 289
configuring 26
default parameters 103
lag_limit parameter 111, 284
logreuse parameter 111, 284, 290
logstdout parameter 111, 285, 290
memory_limit parameter 112, 285, 290
monitor_interval parameter 112, 285, 290
monitor_limit parameter 112, 285, 290
operating 103, 288
prune_interval parameter 112, 285, 290
pruning 288
pwdfile parameter 285
reinitializing 119, 288
resuming 119, 288
retention_limit parameter 113, 285, 290
setting up 25
sleep_interval parameter 114, 285, 291
starting 107, 282, 511
startmode parameter 114, 286
status of 288
stopping 118, 288
Capture program *(continued)*
for UNIX *(continued)*
suspending 118, 288
term parameter 115, 286, 291
trace_limit parameter 115, 287, 291
warm start parameters 114, 286
for Windows
add_partition parameter 108, 283
autoprune parameter 108, 284, 289
autostop parameter 109, 284, 289
binding 26
capture_path parameter 109, 283
capture_schema parameter 110, 283
capture_server parameter 110, 283
changing parameters 288
checking status 165
cold start parameters 114, 286
commit_interval parameter 110, 284, 289
configuring 26
default parameters 103
lag_limit parameter 111, 284
logreuse parameter 111, 284, 290
logstdout parameter 111, 285, 290
memory_limit parameter 112, 285, 290
monitor_interval parameter 112, 285, 290
monitor_limit parameter 112, 285, 290
operating 103, 288
prune_interval parameter 112, 285, 290
pruning 288
pwdfile parameter 285
reinitializing 119, 288
resuming 119, 288
retention_limit parameter 113, 285, 290
setting up 25
sleep_interval parameter 114, 285, 291
starting 107, 282, 511
startmode parameter 114, 286
status of 288
stopping 118, 288
suspending 118, 288
term parameter 115, 286, 291
trace_limit parameter 115, 287, 291
warm start parameters 114, 286
for z/OS
add_partition parameter 108, 283
autoprune parameter 108, 284, 289
autostop parameter 109, 284, 289
capture_path parameter 109, 283
capture_schema parameter 110, 283
capture_server parameter 110, 283
changing parameters 288
checking status 165
cold start parameters 114, 286

Capture program *(continued)*
for z/OS *(continued)*
commit_interval parameter 110, 284, 289
default parameters 103
lag_limit parameter 111, 284
logreuse parameter 111, 284, 290
logstdout parameter 111
memory_limit parameter 112, 285, 290
monitor_interval parameter 112, 285, 290
monitor_limit parameter 112, 285, 290
operating 103, 288
prune_interval parameter 112, 285, 290
pruning 288
pwdfile parameter 285
reinitializing 119, 288
resuming 119, 288
retention_limit parameter 113, 285, 290
setting up 30
sleep_interval parameter 114, 285, 291
starting 107, 282
startmode parameter 114, 286
status of 288
stopping 118, 288
suspending 118, 288
term parameter 115, 286, 291
trace_limit parameter 115, 287, 291
warm start parameters 114, 286
latency analysis 169
memory used by 3
messages 168
printing 168
operating 238
performance data 167
running more than one 24
scheduling 413
setting defaults for parameters 105
setting environment variables 25
signals 196
throughput analysis 168
user ID 18
where to start it 110
Capture schemas
changing 179
naming rules 269
using multiple 24
Capture schemas (CAPSCHEMAS) table 434
Capture signals 196
Capture trace (CAPTRACE) table
pruning 214
structure 440
Capture triggers
authorization requirements 19
communicating with
Apply program 415, 417
Replication Center 415
conflicts with pre-existing triggers 11
names of 11
planning 10
capture_path parameter 109, 283
capture_schema parameter 110, 283
capture_server parameter 110, 283
catalog tables, registering 35
CCD (consistent-change-data) tables
adding UOW columns 73
external
multi-tier replication 75
internal
multiple targets 74
locks on 11
non-DB2 relational data sources
using CCD tables 37
nonrelational data sources
maintaining CCD tables 55
using CCD tables 35
replication sources 75
structure
Capture control server 441
target server 502
usage
history or audit 73
multi-tier replication 75
CD (change-data) tables
for joins 52
for views 52
pruning 213
storage requirements 8
structure 442
summarizing contents 73
triggers on 94
CD (change-data) views 52
change aggregate tables
definition 69
structure 501
usage 73
change capture
enabling 230
change-capture replication
description 40
registration option 39
change-data (CD) tables
pruning 213
storage requirements 8
structure 442
summarizing contents 73
changing Capture parameters
for OS/400 355
for UNIX 288
for Windows 288
for z/OS 288
character large object (CLOB)
replication considerations 85
CHGDPRCAPA command 355
CHGJRN command 32
CLNUPITV parameter 395
CLOB (character large object)
replication considerations 85
code pages
compatible 12
DB2CODEPAGE environment variable 12
translation 11
cold start, Capture program
for OS/400 394, 399
for UNIX 114, 286
for Windows 114, 286
cold start, Capture program *(continued)*
for z/OS 114, 286
preventing 215
cold startmode 114
column (vertical) subsetting
at the source 39
at the target 80
columns
adding to registered source tables 174
after-image 41
available for replication 39
before-image 41
calculated 80
computed 99
defining in target table 80
mapping from sources to targets 81
registering in source table 39
relative record numbers on OS/400 51
renaming 81, 98
subsetting
at the source 39
at the target 80
commit_interval parameter
overview 110
tuning 4
use with asncap command 284
use with asnccmd command 289
compression dictionaries (z/OS) 208
computed columns
CD table 73
creating 99
source table 72
configuration parameters for DB2
APPLHEAPSZ 26
DBHEAP 26
LOGBUFSZ 26
LOGFILSIZ 26
LOGPRIMARY 26
LOGSECOND 26
MAXAPPLS 26
configuring
Apply program
for UNIX 26
for Windows 26
Capture program
for UNIX 26
for Windows 26
connectivity 15
Replication Alert Monitor
for UNIX 27
for Windows 27
Replication Center 221
conflict detection
levels of 50
overview 49
peer-to-peer replication 10
planning 10
requirements 42
update-anywhere replication 10
conflicts
preventing 10
connecting
to iSeries server 16
to z/OS server 16

connectivity
between DB2 operating systems 15, 16
failure recovery for control tables 215
consistent-change-data (CCD) tables
adding UOW columns 73
external
multi-tier replication 75
internal
multiple targets 74
locks on 11
non-DB2 relational data sources
using CCD tables 37
nonrelational data sources
maintaining CCD tables 55
using CCD tables 35
replication sources 75
structure
Capture control server 441
target server 502
usage
history or audit 73
multi-tier replication 75
contact groups 143
contacts
defining 152
description 143
control servers, adding to Replication Center 229
control tables
APPENQ (Apply enqueue) 463
Apply
creating 228
Apply control server 431
APPLY_JOB (Apply job) 464
APPLYTRACE (Apply trace) 467
APPLYTRAIL (Apply trail) 468
APPPARMS (Apply parameters) 465
at Apply control server 463
at Capture control server 433
at Monitor control server 487
authorization requirements for OS/400 30
AUTHTKN (Apply-qualifier cross-reference) 434
CAPENQ (Capture enqueue) 435
CAPMON (Capture monitor)
pruning 214
structure 436
CAPPARMS (Capture parameters)
structure 437
CAPSCHEMAS (Capture schemas) 434
CAPTRACE (Capture trace)
pruning 214
structure 440
Capture
creating 227
Capture server 428
CCD (consistent-change-data)
Capture control server 441
target server 502
CD (change-data) 442
connectivity failure recovery 215
creating
for Apply 228
control tables *(continued)*
creating *(continued)*
for Capture 227
for non-DB2 relational sources 23
for Replication Alert Monitor 228
in IASP groups 23
multiple database operating system 22
multiple database partitions 24
multiple sets 24
on Linux, UNIX, Windows 22
on OS/400 23, 360
on z/OS 22
Replication Alert Monitor 151
dynamic 209
granting authority for OS/400 18, 366
I/O error recovery 215
maintaining 209
Monitor
creating 228
Monitor control server
IBMSNAP_ALERTS 487
IBMSNAP_CONDITIONS 488
IBMSNAP_CONTACTGRP 493
IBMSNAP_CONTACTS 494
IBMSNAP_GROUPS 495
IBMSNAP_MONENQ 495
IBMSNAP_MONPARMS 495
IBMSNAP_MONSERVERS 497
IBMSNAP_MONTRAIL 499
PARTITIONINFO (partition information) 443
profiles 224
PRUNCNTL (pruning control) 444
PRUNE_LOCK (prune lock) 446
PRUNE_SET (prune set) 447
pruning 212
quick reference
Apply control server 431
at a glance 421
Capture server 428
target server 433
rebinding, packages and plans 210
REG_EXT (register extension) 447
REG_SYNCH (register synchronization) 455
REGISTER (register) 449
reorganizing 210
RESTART (restart) 456
revoking authority for OS/400 384
RUNSTATS utility 210
SEQTABLE (sequencing) 458
SIGNAL (signal) 458
static 211
storage requirements 7
SUBS_COLS (subscription columns) 473
SUBS_EVENT (subscription events) 475
SUBS_MEMBR (subscription members) 475
SUBS_SET (subscription sets) 480
SUBS_STMTS (subscription statements) 485
target server 433
UOW (unit-of-work) 461
control_server parameter 125, 272, 277
copying replication configurations 203
copyonce parameter 125, 279
COPYONCE parameter 390
correlation ID 52
creating
control tables
Replication Alert Monitor 151
monitors 154
creating control tables 22
creating subscription sets 232
CRTDPRTBL command 360
CRTJRN command 31
CRTJRNRCV command 31
CTLSVR parameter 388
current receiver size 6, 32
customizing, SQL scripts 101

D

data
advanced subsetting techniques 93
displaying historical 167
manipulating 97
preventing double-deletes 53
retrieving from source tables 216
subsetting
during registration 93
using predicates 95
using triggers on CD tables 94
using views 94
using views to specify predicates 95
transforming
at registration 97
at subscription 98
creating computed columns 99
renaming columns 81, 98
data blocking 61
data consistency 79
Data Links
replication 86
Data Links Manager replication daemon 90
data types
mapping between columns 81
replicating
DATALINK values 86
large objects (LOB) 85
restrictions 85
databases, enabling for change capture 230
DATALINK values
ASNDLCOPY exit routine 88
ASNDLCOPYD file-copy daemon 91
DLFM_ASNCOPYD file-copy daemon 90
replicating 86
restrictions 49, 78
storing updates 45
DB2 Extenders
restrictions 86
DB2 replication
authorization requirements 17
DB2 tables
registering 35

DB2 views
registering 54
db2_subsystem parameter 126, 277
DB2CODEPAGE environment variable 12, 25
DB2DBDFT environment variable 25
DB2INSTANCE environment variable 25
db2rc command 221
DBADM 18
DBCLOB (double-byte character large object)
replication considerations 85
DBHEAP configuration parameter 26
deactivating
registered objects 177
subscription sets 61, 194
deactivating subscription sets 235
defaults
for Apply parameters (Linux, UNIX, Windows, z/OS) 121, 124
for Apply parameters (OS/400) 132
for Capture parameters (Linux, UNIX, Windows, z/OS) 103
for Capture parameters (OS/400) 104, 105
for Capture parameters (UNIX, Windows, z/OS) 108
delay parameter 126, 280
DELAY parameter 390
delete journal receiver exit routine
about 33
registering 34
removing 33
delimiters, in generated SQL scripts 102
diagnostic files
storage 8, 9
differential refresh replication
See change-capture replication
disk space
requirements 5
temporary files 8
display names 410
distinct data types 85
distributed recovery points 199
DLFM_ASNCOPYD file-copy daemon 90
double-byte character large object (DBCLOB)
replication considerations 85
double-deletes 53
DPR registrations (OS/400)
adding 319
removing 380
DSPJRN command 171
dynamic control tables 209

E

e-mail notification, replication 149
editing, SQL scripts 101
EDITPROC clauses
restrictions, compression 85
email_server parameter, Replication Alert Monitor 158
ENDDPRAPY command 361
ENDDPRCAP command 118, 364
ENDJOB command 365
environment variables
Capture program 25
DB2CODEPAGE 12, 25
DB2DBDFT 25
DB2INSTANCE 25
LIBPATH 25
errors
monitor_errors parameter 158
monitoring with alert conditions 143
operational 162
replication
alert conditions, APPLY_ERRORS 146
alert conditions, CAPTURE_ERRORS 146
alert conditions, QAPPLY_ERRORS 146
alert conditions, QCAPTURE_ERRORS 146
SQL 146
errwait parameter 126, 280
event publishing commands
asnslist 306
asntdiff 307
asntrep 316
event-based scheduling 66
events, coordinating 196
existing tables as targets 79
exit routines
ASNDLCOPY 88
ASNDONE
using 133, 134
ASNLOAD
customizing 138
for OS/400 140
for UNIX 136
for Windows 136
for z/OS 137
using 135
delete journal receiver (OS/400) 33
external CCD tables
multi-tier replication 75

F

FIELDPROC clauses
restrictions, compression 85
file-copy daemons
ASNDLCOPYD 91
DLFM_ASNCOPYD 90
files
*.APP.log 127
*.CAP.log 111
*.err 130
*.sqs 129, 130
asndone.smp 134
asnload.ini 139
spill 8
fragmentation
horizontal
at the source 40
at the target 80
peer-to-peer replication 10
update-anywhere replication 10
vertical
at the source 39
at the target 80
FRCFRQ parameter 398
full-refresh copying
Apply for iSeries 52, 389
forcing 237
registration option 39
FULLREFPGM parameter 389

G

gap detection 73
generated SQL scripts 101
global record 450
GRTDPRAUT command
granting privileges to SQL packages 29
syntax 366
GRTOBJAUT command 29

H

heterogeneous replication
registering sources 37
restrictions
aggregate tables 72
CCD tables 41
multi-tier replication 75
update-anywhere 45, 78
history data
CCD tables 73
source data 42
horizontal (row) subsetting
at the source 40
at the target 80

I

I/O error recovery, control tables 215
IASP groups 23
IBMSNAP_ALERTS control table 487
IBMSNAP_CONDITIONS control table 488
IBMSNAP_CONTACTGRP control table 493
IBMSNAP_CONTACTS control table 494
IBMSNAP_GROUPS control table 495
IBMSNAP_MONENQ control table 495
IBMSNAP_MONPARMS control table 495
IBMSNAP_MONSERVERS control table 497
IBMSNAP_MONTRAIL control table 499
IMS data sources
maintaining CCD tables 55
registering 35
using CCD tables 35
IMS DataPropagator 35
inactive subscription sets 61
INACTMSG parameter 389
inamsg parameter 127, 278
Independent Auxiliary Storage Pool (IASP) groups 23
indexes
target tables 82
inner-joins as sources 52

internal CCD tables
multiple targets 74
interval timing 66
invocation parameters
Analyzer
for OS/400 353
for UNIX 273
for Windows 273
Apply program
for OS/400 131, 387
for UNIX 124, 277
for Windows 124, 277
for z/OS 124, 277
Capture program
for OS/400 103, 115, 356, 394
for UNIX 108, 283
for Windows 108, 283
for z/OS 108, 283
Replication Alert Monitor
for UNIX 295
for Windows 295
for z/OS 295
replication commands
for OS/400 321, 329, 344, 360, 361, 364, 366, 374, 376, 380, 381, 383, 385, 387, 394, 401
INZDPRCAP command 374
iSeries server
connecting to 16

J

JOBD parameter 387, 394
JOIN_UOW_CD column 95
joins as sources 52
journal jobs
checking status 166
journal message queues 33
journal receivers
creating for source tables 31
current, size 6
delete journal receiver exit routine 33
maintaining 205
managing 32
retaining 208
system management 32
threshold 32
user management 32
journal signal tables 197
CAPSTOP 202
creating 197
stopping 200
journals
creating 31
creating for source tables 30
default message queue 33
entry types 509
managing 32
QSQJRN journal 30
registering as sources 35
setup 30
starting 31
using 30
using remote journal function 51
JRN parameter 396

L

LAG parameter 398
lag_limit parameter 111, 284
LANG variable
setting 13
large object (LOB)
replication considerations 85
large replication jobs 61
latency
Apply program 170
Capture program 169
launchpad 222
LIBPATH 25
load from cursor function 139
loadxit parameter 127, 278
LOB (large object)
replication considerations 85
update-anywhere restrictions 78
locks
on CCD tables 11
log
planning impact to 11
log records
archived before captured 6
compression dictionaries (z/OS) 208
maintaining 205
multi database partitions 206
retaining 206
LOGBUFSZ configuration parameter 26
LOGFILSIZ configuration parameter 26
logging requirements
DB2 source servers 6
non-DB2 relational source servers 11
target servers 7
logical partitioning keys
description 44
LOGPRIMARY configuration parameter 26
logreuse parameter (for Apply) 127, 278
logreuse parameter (for Capture) 111, 284, 290
LOGSECOND configuration parameter 26
logstdout parameter (for Apply) 127, 278
logstdout parameter (for Capture) 111, 285, 290
LONG VARCHAR data types 85
LONG VARGRAPHIC data types 85

M

manipulating data
at registration 97
at subscription 98
creating computed columns 99
renaming columns 81, 98
mapping
data types between tables 81
source columns to target columns 81
sources to targets 67
master tables (update-anywhere)
overview 77
recapturing changes 45
max_notification_minutes parameter, Replication Alert Monitor 158
max_notifications_per_alert parameter, Replication Alert Monitor 158
MAX_SYNCH_MINUTES, data blocking 61
MAXAPPLS configuration parameter 26
MEMLMT parameter 397
memory
alert conditions
APPLY_MEMORY 146
CAPTURE_MEMORY 146
QAPPLY_MEMORY 146
QCAPTURE_MEMORY 146
Apply program 5
batch jobs 3
Capture program 3
planning 3
reading log records 4
registrations 4
Replication Alert Monitor 5
subscription sets 5
transactions 3
using CAPMON table to tune 4
memory_limit parameter
overview 112
tuning 4
use with asncap command 285
use with asnccmd command 290
merging
subscription sets 189
triggers 11
message queues, for journals 33
messages 168, 169, 171
Microsoft SQL Server
replication restrictions 41
migration
planning 3
mini-cycles 61
Monitor
See Replication Alert Monitor
Monitor control server
adding to Replication Center 229
control tables at 487
IBMSNAP_ALERTS control table 487
IBMSNAP_CONDITIONS control table 488
IBMSNAP_CONTACTGRP control table 493
IBMSNAP_CONTACTS control table 494, 495
IBMSNAP_MONENQ control table 495
IBMSNAP_MONPARMS control table 495
IBMSNAP_MONSERVERS control table 497
IBMSNAP_MONTRAIL control table 499
Monitor control tables
list of 487
Monitor program
messages 171
printing 171
monitor qualifier
replication 143
Monitor qualifiers, naming rules 269
monitor_errors parameter, Replication Alert Monitor 158

monitor_interval parameter (for Capture) 112, 285, 290
monitor_limit parameter 112, 285, 290
  Replication Alert Monitor 158
monitor_path parameter, Replication Alert Monitor 158
monitoring
  for OS/400 171
  historical trends 167
  replication 143, 153
  status of programs 166
MONITV parameter 397
MONLMT parameter 396
multi database partition
  log records 206
multi-tier replication
  defining subscription sets 75
multiple database partitions
  Capture 30
multiple target tables 74

N

names
  Apply qualifier rules 269
  Capture schema rules 269
  display names 410
  for Windows services 270
  Monitor qualifier rules 269
  of Capture triggers 11
  of replication services 409
  subscription sets 184
national language support (NLS) 12
network connectivity 15
nicknames
  for load from cursor function 139
  registering 37
  restrictions
    aggregate tables 72
    multi-tier replication 75
    update-anywhere 45, 78
    with CCD tables 41
NLS (national language support) 12
non-DB2 relational data sources
  locks 11
  registering 37
  restrictions
    aggregate tables 72
    multi-tier replication 75
    update-anywhere 45, 49, 78
  source servers 11
  using CCD tables 37
nonrelational data sources
  maintaining CCD tables 55
  using CCD tables 35
notify parameter 128, 278

O

objects
  changing attributes 174
  deactivating 177
  reactivating 178
  registering 173
  stop capturing changes 177
operating
  Apply program 239, 271
  Capture program 238, 288
  Replication Alert Monitor 153, 239
opt4one parameter 128, 280
OPTSNGSET parameter 391
OS/400 data sources
  with remote journaling 51
overriding attributes (OS/400)
  Capture program 375
OVRDPRCAPA command 375

P

packages, rebinding 210
parameters
  Replication Alert Monitor
    alert_prune_limit 158
    autoprune 158
    default values 157
    description 158
    email_server 158
    max_notification_minutes 158
    max_notifications_per_alert 158
    monitor_errors 158
    monitor_limit 158
    monitor_path 158
    runonce 158
    trace_limit 158
parameters, invocation
  Analyzer
    for OS/400 353
    for UNIX 273
    for Windows 273
  Apply program
    for OS/400 131, 387
    for UNIX 124, 277
    for Windows 124, 277
    for z/OS 124, 277
  Capture program
    for OS/400 356, 394
    for UNIX 108, 283
    for Windows 108, 283
    for z/OS 108, 283
  Replication Alert Monitor
    for UNIX 295
    for Windows 295
    for z/OS 295
  replication commands
    for OS/400 321, 329, 344, 360, 361, 364, 366, 374, 376, 380, 381, 383, 385, 387, 394, 401
partition information (PARTITIONINFO) table 443
Partition information (PARTITIONINFO) table 443
PARTITIONINFO (partition information) table 443
password files
  storing 22
passwords for Replication Center 223
peer-to-peer replication
  conflict detection 10
performance
  tuning 13
planning
  coexistence of triggers 11
planning *(continued)*
  conflict detection 10, 49
  locks on CCD tables 11
  log impact 6, 11
  memory 3
  migration 3
  storage requirements 5
  transaction throughput rates 10
plans, rebinding 210
point-in-time tables
  structure 504
  usage 72
predicates
  defining for target tables 80
  subsetting 95
PREDICATES column 95
prefix, before-image 43
primary keys
  logical partitioning 44
  relative record numbers for OS/400 51
  used as target key 83
printing
  Apply program
    messages 169
  Capture program
    printing 168
  Monitor program
    messages 171
profiles
  control-table 224
  description 224
  source-object 225
  target-object 226
promoting
  registered tables or views 236
  replication configurations 203
  subscription sets 237
PRUNCNTL (pruning control) table 444
prune intervals
  Replication Alert Monitor 161
prune lock (PRUNE_LOCK) table 446
prune set (PRUNE_SET) table 447
prune_interval parameter 112, 285, 290
PRUNE_LOCK (prune lock) table 446
PRUNE_SET (prune set) table 447
pruning
  Apply trace (APPLYTRACE) table 214
  Apply trail (APPLYTRAIL) table 214
  Capture monitor (CAPMON) table 214
  Capture program
    for UNIX 288
    for Windows 288
    for z/OS 288
  Capture trace (CAPTRACE) table 214
  CD (change-data) tables 213
  control tables 212
  signal (SIGNAL) table 214
  UOW (unit-of-work) table 213, 461
pruning control (PRUNCNTL) table 444
pwdfile parameter 128, 277, 285

Q

Q Apply program
  alert conditions 146
Q Capture program
  alert conditions 146
Q replication commands
  asnslist 306
  asntdiff 307
  asntrep 316

R

RCVJRNE command 32
reactivating
  objects 178
  registrations 178
  tables 178
read dependencies 50
rebinding, packages and plans 210
recapturing changes (update-anywhere) 45
receiver size, current 6
recovery points, distributed 199
referential integrity 79
REG_EXT (register extension) table 447
REG_SYNCH (register synchronization) table 455
register (REGISTER) table 449
REGISTER (register) table 449
register extension (REG_EXT) table 447
register synchronization (REG_SYNCH) table 455
registering
  DB2 tables 35
  IMS data sources 35
  non-DB2 relational data sources 37
  objects 173
  options for sources
    after-image columns 41
    before-image columns 41
    before-image prefix 43
    change-capture replication 39
    column (vertical) subsetting 39
    conflict detection 49
    full-refresh copying 39
    recapturing changes (update-anywhere) 45
    relative record numbers 51
    row (horizontal) subsetting 40
    stop Capture on error 43
    updates as deletes and inserts 44
    using remote journals 51
  tables 173
  views
    overview 52, 54
    procedure 173
registering sources 231
registrations
  adding 319
  adding columns 174
  attributes, changing 174
  deactivating 177
  reactivating 178
  removing 179, 380
  stop capturing changes 177
registry variables
  DB2CODEPAGE 12, 25
  DB2DBDFT 25
  DB2INSTANCE 25
reinitializing
  Replication Alert Monitor 156
reinitializing Capture program
  for UNIX 119
  for Windows 119
  for z/OS 119
relative record numbers
  as primary key for OS/400 51
  support for OS/400 51
  used as target key 83
relative timing 66
remote journals as sources 51
remote source tables 51
renaming columns 81, 98
reorganizing
  control tables 210
replica tables
  defining read-write targets 77
  definition 69
  recapturing changes 45
  structure 505
Replication Alert Monitor
  alert conditions
    e-mail notifications 149
    events 143
    list 146
    overview 145
    selecting 155
    status 143
    thresholds 143
  alerts 143
  authorization requirements 21
  communicating with
    Apply program 419
    Capture 419
    Replication Center 418
  contact groups 143
  contacts 143
  control tables
    IBMSNAP_ALERTS 487
    IBMSNAP_CONDITIONS 488
    IBMSNAP_CONTACTGRP 493
    IBMSNAP_CONTACTS 494
    IBMSNAP_GROUPS 495
    IBMSNAP_MONENQ 495
    IBMSNAP_MONPARMS 495
    IBMSNAP_MONSERVERS 497
    IBMSNAP_MONTRAIL 499
  control tables, creating 151
  defining contact information 152
  description 143
  for UNIX
    binding 27
    checking status 165
    starting 511
  for Windows
    binding 27
    checking status 165
    starting 511
  for z/OS
    checking status 165
  memory usage 5
  monitoring replication, overview 143
Replication Alert Monitor *(continued)*
  monitors
    creating 154
    reinitializing 156
  operating 153, 239
  parameters
    alert_prune_limit 158
    autoprune 158
    default values 157
    descriptions 158
    email_server 158
    how often the Replication Alert Monitor runs 161
    max_notification_minutes 158
    max_notifications_per_alert 158
    monitor_errors 158
    monitor_interval 158
    monitor_limit 158
    monitor_path 158
    notification criteria for alert conditions 162
    notification criteria for operational errors 162
    prune intervals for data 161
    runonce 158
    setting 160
    trace_limit 158
  reinitializing 156
  scheduling 413
  setting up 151
  starting 156
  stopping 163
Replication Analyzer
  for OS/400
    creating SQL packages 29
    invocation parameters 353
  for UNIX, invocation parameters 273
  for Windows, invocation parameters 273
Replication Center
  activating subscription sets 235
  adding servers 229
  communicating with
    Apply program 415
    Capture program 415
    Capture triggers 415
    Replication Alert Monitor 418
  configuring 221
  connectivity 15
  control tables 227
  control-table profiles 224
  creating subscription sets 232
  deactivating subscription sets 235
  deleting definitions 238
  description 219
  enabling databases for change capture 230
  forcing full refresh 237
  launchpad 222
  operating Apply program 239
  operating Capture program 238
  operating Replication Alert Monitor 239
  profiles 224
  promote functions 203
  promoting registered tables or views 236

Replication Center *(continued)*
promoting subscription sets 237
registering sources 231
removing definitions 238
source-object profiles 225
starting 221
target-object profiles 226
user IDs and passwords 223
replication commands
$TA JES2
Apply for z/OS 413
Capture for z/OS 413
ADDJOBSCDE 414
asnslist 306
asntdiff 307
asntrep 316
AT 413
AT NetView
Apply for z/OS 413
Capture for z/OS 413
backup database 26
CRTJRNRCV 31
db2rc 221
DSPJRN 171
for OS/400
ADDDPRREG 319
ADDDPRSUB 327
ADDDPRSUBM 343
ADDEXITPGM 34
ANZDPR 352
ANZDPRJRN 33
CHGDPRCAPA 355
CHGJRN 32
CRTDPRTBL 360
CRTJRN 31
ENDDPRAPY 361
ENDDPRCAP 118, 364
ENDJOB 365
GRTDPRAUT 29, 366
GRTOBJAUT 29
INZDPRCAP 374
OVRDPRCAPA 375
RCVJRNE 32
RMVDPRREG 380
RMVDPRSUB 381
RMVDPRSUBM 383
RMVEXITPGM 33
RVKDPRAUT 384
SBMJOB 414
STRDPRAPY 132, 386
STRDPRCAP 393
STRJRNPF 31
WRKDPRTRC 400
WRKJOB 166
WRKREGINF 34
WRKSBMJOB 166
WRKSBSJOB 166
for UNIX
asnacmd 271
asnanalyze 273
asnapply 276
asncap 282
asnccmd 288
for Windows
asnacmd 271
asnanalyze 273
asnapply 276
replication commands *(continued)*
for Windows *(continued)*
asncap 282
asnccmd 288
for z/OS
asnacmd 271
asnapply 276
asncap 282
asnccmd 288
update database configuration 26
replication environments
copying 203
replication events coordination 196
replication services
creating 409
dropping 410
listing 306
names 409
operating 410
replication sources
CCD (consistent-change-data) tables 75
joins 52
maintaining CCD tables 55
mapping to targets 67
registering
columns 39
DB2 tables 35
IMS data sources 35
non-DB2 relational data sources 37
rows 40
views 54
subscribing to 59
restart (RESTART) table 456
RESTART (restart) table 456
RESTART parameter 394
restrictions
abstract data types 85
ASCII tables 507
CCD tables 78
column names, limits 42
data types 85
DATALINK values 49, 78
DB2 Extenders large objects 86
distinct data types 85
EDITPROC clauses 85
existing target tables 79
FIELDPROC clauses 85
heterogeneous replication 41, 75, 78
LOB data types 78
LONG columns in Oracle tables 85
LONG VARCHAR data types 85
LONG VARGRAPHIC data types 85
Microsoft SQL Server 41
non-DB2 relational data sources 45, 49
Oracle sources 85
spatial data types 85
stored procedures 98
Sybase 41
Unicode tables 507
user-defined data types 85
VALIDPROC clauses 85
views 54
WHERE clause 81
resuming
Capture program
for UNIX 119, 288
for Windows 119, 288
for z/OS 119, 288
RETAIN parameter 397
retention_limit parameter 113, 285, 290
RMVDPRREG command 380
RMVDPRSUB command 381
RMVDPRSUBM command 383
RMVEXITPGM command 33
roll-forward recovery 26
row (horizontal) subsetting
at the source 40
at the target 80
row-capture rules 40
ROWID 86
rows
available for replication 40
defining in target table 80
registering in source table 40
subsetting
at the source 40
at the target 80
RRN 51
RTYWAIT parameter 390
run-time processing 65, 98
running, SQL scripts 101
runonce parameter, Replication Alert Monitor 158
RUNSTATS utility 210
RVKDPRAUT command 384

S

SBMJOB command 414
scenario
create Apply control tables 249
create Apply password file 254
create Capture control tables 246
create contacts 262
create Monitor control tables 260
create subscription set 250
enable source database for replication 246
monitoring replication 260
operations 256
planning 244
prerequisites 243
register a source 247
replicate data 255
select alert conditions for Apply program 263
select alert conditions for Capture program 262
setup 246
start the Replication Alert Monitor 264
status for Apply program 258
status for Capture program 257
stop Capture and Apply programs 259
update source table 257
scheduling
replication programs 413
subscription sets 65, 66

schemas
changing 179
naming rules 269
SCM (Service Control Manager)
creating replication services 409
dropping replication services 410
naming replication services 409
operating replication services 410
SEQTABLE (sequencing) table 458
sequencing (SEQTABLE) table 458
servers
adding to Replication Center 229
Service Control Manager (SCM)
creating replication services 409
dropping replication services 410
naming replication services 409
operating replication services 410
services
Windows SCM 409
setting environment variables
Capture program 25
setting up
Apply programs
for OS/400 28
for UNIX 25
for Windows 25
Capture programs
for OS/400 28
for UNIX 25
for Windows 25
journals 30
Replication Alert Monitor 27, 151
signal (SIGNAL) table
pruning 214
structure 458
SIGNAL (signal) table
pruning 214
structure 458
signals
CAPSTART 201
CAPSTOP 202
setting distributed recovery points 199
STOP 198, 199
USER 196
sleep parameter 128, 279
sleep_interval parameter 114, 285, 291
source logs, maintaining 205
source servers
DB2
log impact 6
non-DB2 relational
log impact 11
source systems, maintaining 205
source tables
adding columns 174
creating journals for 30
maintaining 205
retrieving lost data 216
sources
CCD (consistent-change-data) tables 75
maintaining CCD tables 55
mapping to targets 67
profiles 225
promoting 236
sources *(continued)*
registering
DB2 tables 35
IMS data sources 35
non-DB2 relational 37
Replication Center 231
views 52, 54
registering columns 39
registering rows 40
registration options
after-image columns 41
before-image columns 41
before-image prefix 43
change-capture replication 39
column (vertical) subsetting 39
conflict detection 49
full-refresh copying 39
recapturing changes (update-anywhere) 45
relative record numbers 51
row (horizontal) subsetting 40
stop Capture on error 43
updates as deletes and inserts 44
using remote journals 51
subscribing to 59
spatial data types 85
special data types
replicating
DATALINK values 86
large objects (LOB) 85
spill files
storage for Apply 9
storage for Capture 8
storage for diagnostic files 8
spillfile parameter 129, 281
splitting
subscription sets 186
SQL files, editing 101
SQL packages
creating for Apply program 29
creating for Capture program 28, 29
creating for Replication Analyzer 29
SQL scripts 101
SQL statements
defining for subscription set 65
run-time processing 98
sqlerrcontinue parameter 129, 281
staged replication 76
staging data 75
starting
Apply program
for OS/400 131, 386
for UNIX 123, 276, 511
for Windows 123, 276, 511
for z/OS 123, 276
Capture program
for OS/400 115, 393
for UNIX 107, 282, 511
for Windows 107, 282, 511
for z/OS 107, 282
using Windows services 409
Replication Alert Monitor
for UNIX 511
for Windows 511
various methods 156
starting Replication Center 221
startmode parameter 114, 286
static control tables 211
status
Apply program 165, 166
Capture program 165, 166
journal jobs 166
Replication Alert Monitor 165
stop Capture on error option 43
stop capturing changes 177
STOP signals 198, 199
stopping
Apply program
for OS/400 133, 361
for UNIX 133, 271
for Windows 133, 271
for z/OS 133, 271
Capture program
for OS/400 118, 364
for UNIX 118, 288
for Windows 118, 288
for z/OS 118, 288
Replication Alert Monitor 163
storage
Apply diagnostic files 9
Apply spill files 9
Capture diagnostic files 8
Capture spill files 8
CD table 8
control tables 7
database log and journal data 6
diagnostic files 8
requirements 5
target tables 7
temporary files 8
UOW table 8
stored procedures
defining for subscription set 65
manipulating data 98
STRDPRAPY command 132, 386
STRDPRCAP command 393
STRJRNPF command 31
SUBNFYPGM parameter 389
SUBS_COLS (subscription columns) table 473
SUBS_EVENT (subscription events) table
posting events 66
structure 475
SUBS_MEMBR (subscription members) table 139, 475
SUBS_SET (subscription sets) table 480
SUBS_STMTS (subscription statements) table 485
subscribing to sources 59
subscription columns (SUBS_COLS) table 473
subscription cycle 61
subscription events (SUBS_EVENT) table
posting events 66
structure 475
subscription members (SUBS_MEMBR) table 139, 475
subscription sets
activating 235
activation level 61
adding 327
adding members 67, 182
changing
Apply qualifiers 192

subscription sets *(continued)*
changing *(continued)*
attributes 183
names 184
columns 80
creating 59, 182, 232
data consistency 79
deactivating 194, 235
disabling members 183
enabling members 183
merging 189
mini-cycles 61
multi-tier replication 75
number of Apply qualifiers 58
processing mode 64
promoting 237
referential integrity 79
removing 195, 381
rows 80
run-time processing statements 98
scheduling
event-based 66
time-based 66
splitting 186
SQL statements 65
stored procedures 65
update-anywhere replication 77
subscription sets (SUBS_SET) table 480
subscription statements (SUBS_STMTS) table 485
subscription-set members
adding 67, 182, 343
applying subset of columns 80
applying subset of rows 80
defining target key 82
disabling 183
enabling 183
mapping between columns 81
mapping data types 81
multi-tier replication 75
number per subscription set 58
removing 383
selecting target types 69
update-anywhere replication 77
subsetting
advanced techniques
during registration 93
using predicates 95
using triggers on CD tables 94
using views 94
columns at target 80
registered columns 39
registered rows of changes 40
rows of changes at target 80
suspending
Capture program
for UNIX 118, 288
for Windows 118, 288
for z/OS 118, 288
Sybase
replication restrictions 41
SYSADM 18
system change journal management 32
system commands
asnslist 306
asntdiff 307
asntrep 316

T

table differencing utility 307
table repair utility 316
table structures 421
table-mode processing 7, 64
tables
adding columns 174
APPENQ (Apply enqueue) 463
APPLY_JOB (Apply job) 464
APPLYTRACE (Apply trace) 467
APPLYTRAIL (Apply trail) 468
APPPARMS (Apply parameters) 465
at Apply control server 463
at Capture control server 433
at Monitor control server 487
at target server 501
AUTHTKN (Apply-qualifier cross-reference) 434
base aggregate 501
CAPENQ (Capture enqueue) 435
CAPMON (Capture monitor) 214, 436
CAPPARMS (Capture parameters) 437
CAPSCHEMAS (Capture schemas) 434
CAPTRACE (Capture trace) 214, 440
CCD (consistent-change-data)
Capture control server 441
target server 502
CD (change-data) 442
change aggregate 501
changing attributes 174
conflict detection for 10
control tables
connectivity failure recovery 215
creating 22
dynamic 209
I/O error recovery 215
maintaining 209
pruning 212
reorganizing 210
RUNSTATS utility 210
static 211
deactivating 177
IBMSNAP_ALERTS 487
IBMSNAP_CONDITIONS 488
IBMSNAP_CONTACTGRP 493
IBMSNAP_CONTACTS 494
IBMSNAP_GROUPS 495
IBMSNAP_MONENQ 495
IBMSNAP_MONPARMS 495
IBMSNAP_MONSERVERS 497
IBMSNAP_MONTRAIL 499
maintaining CCD tables 55
PARTITIONINFO (partition information) 443
PARTITIONINFO (partitioninfo) 443
point-in-time 504
PRUNCNTL (pruning control) 444
PRUNE_LOCK (prune lock) 446
PRUNE_SET (prune set) 447
reactivating 178
REG_EXT (register extension) 447
REG_SYNCH (register synchronization) 455
REGISTER (register) 449
tables *(continued)*
registering
DB2 35
non-DB2 relational 37
procedure 173
removing registrations 179
replica 10, 505
RESTART (restart) 456
SEQTABLE (sequencing) 458
SIGNAL (signal) 458
stop capturing changes 177
structures 421
SUBS_COLS (subscription columns) 473
SUBS_EVENT (subscription events) 475
SUBS_MEMBR (subscription members) 139, 475
SUBS_SET (subscription sets) 480
SUBS_STMTS (subscription statements) 485
target tables
See also target tables
maintaining 216
UOW (unit-of-work) 461
user copy 505
target indexes 82
target keys 82
target servers
log impact 7
tables at 501
target tables
applying subset of columns 80
applying subset of rows 80
base aggregate
definition 69
structure 501
usage 72
CCD (consistent-change-data)
overview 69
structure 502
change aggregate
definition 69
structure 501
usage 73
defining columns 80
defining rows 80
defining target key 82
fragmenting 80
list of 501
maintaining 216
mapping to sources 67
new columns for 99
point-in-time
definition 69
structure 504
usage 72
replica
conflict detection for 10
definition 69
structure 505
usage 77
storage requirements 7
table structures, quick reference 433
user copy
definition 69
structure 505

target tables *(continued)*
user copy *(continued)*
usage 72
user defined 71, 79
target-key columns
updating 83
targets
forcing full refresh 237
profiles 226
term parameter (for Apply) 130, 280
term parameter (for Capture) 115, 286, 291
termination characters, in generated SQL scripts 102
three-tier replication configuration 76
throughput
Apply program 170
Capture program 168
throughput rates
Capture triggers 10
time-based scheduling 66
tips
checking if Apply processed a set successfully 130
deleting rows from the Apply trail table 131
estimating use of space 6
using sleep versus copyonce parameters 126, 129
using stored procedures for additional processing of sets 133
using stored procedures with ASNDONE 134
verifying a service is set up correctly 409
verifying change capture began 107
trace facility
for OS/400 400
TRACE parameter 388
trace_limit parameter
overview 115
Replication Alert Monitor 158
use with asncap command 287
use with asnccmd command 291
use with asnmon command 297
transaction throughput rates
Capture triggers 10
transaction-mode processing 7, 64
transactions
memory used by 3
transforming data
at registration 97
at subscription 98
creating computed columns 99
renaming columns 81, 98
translating data 12
TRCLMT parameter 396
triggers
capturing data 10
merging 11
on CD tables 94
suppressing data capture 94
trlreuse parameter 130, 279
TRLREUSE parameter 391
troubleshooting commands
WRKDPRTRC 400
tuning
commit_interval parameter 4
memory_limit parameter 4
performance 13

U

Unicode tables 507
unit-of-work (UOW) table
columns in CCD tables 73
pruning 213, 461
storage requirements 8
structure 461
UOW (unit-of-work) table
columns in CCD tables 73
pruning 213, 461
storage requirements 8
structure 461
UOW_CD_PREDICATES column 95
update database configuration command 26
update-anywhere replication
conflict detection
overview 49
planning for 10
requirements 42, 49
defining subscription sets 77
fragmentation for 10
recapturing changes 45
updated primary key columns 44
updates
as deletes and inserts 44
conflicts 49
user copy table
definition 69
structure 505
usage 72
user IDs
authorization 18
for Apply program 19
for Capture program 18
for Capture triggers 19
for Replication Alert Monitor 21
password files 22
user IDs for Replication Center 223
USER parameter 387
USER signals 196
user-defined data types 85
user-defined tables 71, 79
utilities
table differencing 307
table repair 316

V

VALIDPROC clauses 85
vertical (column) subsetting
at the source 39
at the target 80
views
changing attributes 174
registering
as sources 54
overview 52
procedure 173
restrictions 52, 54
views *(continued)*
using correlation ID 52

W

WAIT parameter 395
warm start, Capture program
for OS/400 394, 399
for UNIX 114, 286
for Windows 114, 286
for z/OS 114, 286
warmns startmode 114
warmsa startmode 114
warmsi startmode 114
WHERE clause
PREDICATES column restriction 95
row subsets 80
Windows Service Control Manager (SCM) 409
asnslist command 306
listing replication services 306
Windows services names 270
work management objects 32
WRKDPRTRC command 400
WRKJOB command 166
WRKREGINF command 34
WRKSBMJOB command 166
WRKSBSJOB command 166

Z

z/OS server
connecting to 16

Accessibility

Accessibility features help users with physical disabilities, such as restricted mobility or limited vision, to use software products successfully. The following list specifies the major accessibility features in DB2® Version 8 products:

- All DB2 functionality is available using the keyboard for navigation instead of the mouse. For more information, see "Keyboard input and navigation."
- You can customize the size and color of the fonts on DB2 user interfaces. For more information, see "Accessible display."
- DB2 products support accessibility applications that use the Java™ Accessibility API. For more information, see "Compatibility with assistive technologies" on page 540.
- DB2 documentation is provided in an accessible format. For more information, see "Accessible documentation" on page 540.

Keyboard input and navigation

Keyboard input

You can operate the DB2 tools using only the keyboard. You can use keys or key combinations to perform operations that can also be done using a mouse. Standard operating system keystrokes are used for standard operating system operations.

For more information about using keys or key combinations to perform operations, see Keyboard shortcuts and accelerators: Common GUI help.

Keyboard navigation

You can navigate the DB2 tools user interface using keys or key combinations.

For more information about using keys or key combinations to navigate the DB2 Tools, see Keyboard shortcuts and accelerators: Common GUI help.

Keyboard focus

In UNIX® operating systems, the area of the active window where your keystrokes will have an effect is highlighted.

Accessible display

The DB2 tools have features that improve accessibility for users with low vision or other visual impairments. These accessibility enhancements include support for customizable font properties.

Font settings

You can select the color, size, and font for the text in menus and dialog windows, using the Tools Settings notebook.

For more information about specifying font settings, see Changing the fonts for menus and text: Common GUI help.

Non-dependence on color

You do not need to distinguish between colors in order to use any of the functions in this product.

Compatibility with assistive technologies

The DB2 tools interfaces support the Java Accessibility API, which enables you to use screen readers and other assistive technologies with DB2 products.

Accessible documentation

Documentation for DB2 is provided in XHTML 1.0 format, which is viewable in most Web browsers. XHTML allows you to view documentation according to the display preferences set in your browser. It also allows you to use screen readers and other assistive technologies.

Syntax diagrams are provided in dotted decimal format. This format is available only if you are accessing the online documentation using a screen-reader.

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

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

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

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

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

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document 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 measurements 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 may contain 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 may contain sample application programs, in source language, which illustrate 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.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _*enter the year or years*_. All rights reserved.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM
AS/400
DataPropagator
DB2
iSeries
MVS
OS/390
OS/400
z/OS

The following terms are trademarks or registered trademarks of other companies:

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

Contacting IBM

To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

To learn about available service options, call one of the following numbers:

- In the United States: 1-888-426-4343
- In Canada: 1-800-465-9600

To locate an IBM office in your country or region, see the IBM Directory of Worldwide Contacts on the Web at www.ibm.com/planetwide.

Product information

Information about DB2 Information Integrator is available by telephone or on the Web.

If you live in the United States, you can call one of the following numbers:

- To order products or to obtain general information: 1-800-IBM-CALL (1-800-426-2255)
- To order publications: 1-800-879-2755

On the Web, go to www.ibm.com/software/data/integration/db2ii/support.html. This site contains the latest information about:

- The technical library
- Ordering books
- Client downloads
- Newsgroups
- Fix packs
- News
- Links to Web resources

Comments on the documentation

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 Information Integrator documentation. You can use any of the following methods to provide comments:

- Send your comments using the online readers' comment form at www.ibm.com/software/data/rcf.
- Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the book (if applicable). If you are commenting on specific text, please include the location of the text (for example, a title, a table number, or a page number).

Printed in USA

SC27-1121-02

Spine information:

IBM DB2 Information Integrator SQL Replication Guide and Reference Version 8.2