



SQL Replication Guide and Reference



SQL Replication Guide and Reference

Note

Before using this information and the product that it supports, be sure to read the general information under “Notices and trademarks” on page 467.

Contents

Chapter 1. Planning for SQL replication 1

Migration planning	1
Memory planning	1
Memory used by the Capture program	1
Memory used by the Apply program	3
Storage planning	3
Log impact for DB2 source servers	3
Log impact for target servers	4
Storage requirements of target tables and control tables	4
Space requirements for spill files for the Capture program	6
Space requirements for spill files for the Apply program	6
Space requirements for diagnostic log files (Linux, UNIX, Windows, z/OS)	7
Conflict detection planning	7
Non-DB2 relational source planning	8
Transaction throughput rates for Capture triggers	8
Log impact for non-DB2 relational source servers	8
Coexistence of preexisting triggers with Capture triggers	8
Locks for Oracle source servers	9
Code page translation planning	9
Replication for data between databases with compatible code pages	9
Configure national language support (NLS) for replication	9
Replication planning for DB2 UDB for z/OS	10
Performance tuning	11

Chapter 2. Setting up user IDs and passwords for SQL replication 13

Authorization requirements for administration	13
Authorization requirements for the Capture program	14
Authorization requirements for the Apply program	15
Authorization requirements for Capture triggers on non-DB2 relational databases	16
Storing user IDs and passwords for SQL replication (Linux, UNIX, Windows)	17

Chapter 3. Configuring servers for SQL replication 19

Connectivity requirements for SQL replication	19
Connecting to iSeries servers from Windows	19
Connecting to non-DB2 relational servers	20
Creating control tables for SQL replication	20
Creating control tables for SQL replication	21
Creating control tables (iSeries)	22
Creating control tables for non-DB2 relational sources	22
Creating multiple sets of Capture control tables	23
Capture control tables on multiple database partitions	23

Setting up the replication programs	24
Setting up the replication programs (Linux, UNIX, Windows)	24
Creating SQL packages to use with remote systems (iSeries)	26
Setting up the replication programs (z/OS)	28
Capture for multiple database partitions	28
Setting up journals (iSeries)	28

Chapter 4. Registering tables and views as SQL replication sources 33

Registering DB2 tables as sources	33
Registering non-DB2 relational tables as sources	35
Registration options for source tables	36
Registering a subset of columns (vertical subsetting)	37
Change-capture replication and full-refresh copying	37
After-image columns and before-image columns	38
Before-image prefix	41
Stop the Capture program on error	41
Options for how the Capture program stores updates	42
Preventing the recapture of changes (update-anywhere replication)	42
Options for conflict detection (update-anywhere replication)	46
Registering tables that use remote journaling (iSeries)	47
Using relative record numbers (RRN) instead of primary keys (iSeries)	48
How views behave as replication sources	49
Views over a single table	49
Views over a join of two or more tables	49
Registering views of tables as sources	51
Maintaining CCD tables as sources (IMS)	52

Chapter 5. Subscribing to sources for SQL replication 53

Planning how to group sources and targets	53
Planning the number of subscription-set members	53
Planning the number of subscription sets per Apply qualifier	54
Creating subscription sets	55
Processing options for subscription sets	57
Specifying whether the subscription set is active	57
Specifying how many minutes worth of data the Apply program retrieves	58
Load options for target tables with referential integrity	59
Specifying how the Apply program replicates changes for subscription set members	60
Defining SQL statements or stored procedures for the subscription set	61

Options for scheduling replication of subscription sets	61
Scheduling the subscription set	63
Creating subscription-set members	63
Target table types	65
Common properties for all target table types	75

Chapter 6. Replicating special data types in SQL replication 81

General data restrictions for replication	81
Large object data types	82

Chapter 7. Subsetting data in an SQL replication environment 85

Subsetting data during registration	85
Subsetting source data using views	85
Defining triggers on CD tables to prevent specific rows from being captured	86
Subsetting data during subscription	87

Chapter 8. Manipulating data in an SQL replication environment 89

Enhancing data using stored procedures or SQL statements	90
Mapping source and target columns that have different names	91
Creating computed columns	91

Chapter 9. Operating the Capture program for SQL replication 93

Starting the Capture program (Linux, UNIX, Windows, and z/OS)	93
Starting the Capture program (iSeries)	94
Default operating parameters for the Capture program	95
Descriptions of Capture operating parameters	97
Methods of changing Capture parameters	104
Altering the behavior of a running Capture program	106
Changing saved operating parameters in the IBMSNAP_CAPPARMS table	107
Stopping the Capture program	108
Reinitializing Capture	108
Suspending the Capture program (Linux, UNIX, Windows, z/OS)	109
Resuming Capture (Linux, UNIX, Windows, z/OS)	109

Chapter 10. Operating the Apply program for SQL replication 111

Starting the Apply program (Linux, UNIX, Windows, z/OS)	111
Starting an Apply program (iSeries)	112
Default operating parameters for the Apply program	113
Descriptions of Apply operating parameters	115
Methods of changing Apply operating parameters	122
Changing saved Apply parameters in the IBMSNAP_APPPARMS table (Linux, UNIX, Windows, z/OS)	123

Stopping the Apply program	123
Modifying the ASNDONE exit routine (Linux, UNIX, Windows, z/OS)	123
Modifying the ASNDONE exit routine (iSeries)	124
Refreshing target tables using the ASNLOAD exit routine	126
Refreshing target tables with the ASNLOAD exit routine (Linux, UNIX, Windows)	126
Refreshing target tables with the ASNLOAD exit routine (z/OS)	128
Customizing ASNLOAD exit behavior (Linux, UNIX, Windows, z/OS)	129
Refreshing target tables with the ASNLOAD exit routine (iSeries)	131

Chapter 11. Operating the replication programs (z/OS) 133

Using system-started tasks to operate the replication programs	133
Using JCL to operate replication programs	133
Starting the Apply program on z/OS with JCL	134
Starting the Capture program on z/OS with JCL	135
Using MVS Automatic Restart Manager (ARM) to automatically restart replication	135
Migrating your replication environment to data-sharing mode (z/OS)	136

Chapter 12. Changing an SQL replication environment 139

Registering new objects	139
Changing registration attributes for registered objects	140
Adding columns to source tables	140
Stop capturing changes for registered objects	142
Making registrations eligible for reactivation	143
Removing registrations	144
Changing Capture schemas	145
Creating new subscription sets	146
Adding new subscription-set members to existing subscription sets	147
Disabling subscription-set members from existing subscription sets	148
Enabling subscription-set members to existing subscription sets	148
Changing properties of subscription sets	149
Changing subscription set names	150
Splitting a subscription set	151
Merging subscription sets	154
Changing Apply qualifiers of subscription sets	157
Deactivating subscription sets	158
Removing subscription sets	160
Coordinating replication events with database application events	161
Setting an event END_SYNCHPOINT using the USER type signal	161
When to use the Capture CMD STOP signal	162
Performing a CAPSTART handshake signal outside of the Apply program	165
Performing a CAPSTOP signal	166
Adjusting for Daylight Savings Time (iSeries)	167

Options for promoting your replication configuration to another system	167
--	-----

Chapter 13. Maintaining a SQL replication environment 171

Maintaining source systems	171
Access to source tables and views	171
Source logs and journal receivers	171
Maintaining control tables	175
The RUNSTATS utility for SQL replication (Linux, UNIX, Windows, z/OS)	175
Rebinding packages and plans (Linux, UNIX, Windows, z/OS)	175
Reorganizing your control tables	176
Pruning dynamic control tables maintained by the Capture programs (Linux, UNIX, Windows, z/OS)	176
CD and UOW table pruning	177
Recommendations for pruning other dynamic control tables	178
Preventing replication failures and recovering from errors	179
Maintaining your target tables.	181

Chapter 14. Detecting and repairing differences between source and target tables. 183

Table difference utility (asntdiff)	183
Table repair utility (asntrep)	186

Chapter 15. Monitoring replication with the Replication Alert Monitor . . 189

Replication Alert Monitor	189
Alert conditions and notifications for the Replication Alert Monitor	192
Alert conditions for the Replication Alert Monitor	192
E-mail notifications for replication alert conditions	195
Sending alerts to the z/OS console	196
The ASNMAIL exit routine for sending alerts in replication	196
Setting up the Replication Alert Monitor	197
Memory used by the Replication Alert Monitor	197
Authorization requirements for the Replication Alert Monitor	198
Optional: Binding the Replication Alert Monitor program packages (Linux, UNIX, Windows)	198
Creating control tables for the Replication Alert Monitor	199
Defining contact information for the Replication Alert Monitor	200
Creating monitors for replication or publishing	201
Selecting alert conditions for the Replication Alert Monitor	201
Changing alert conditions for the Replication Alert Monitor	202
Defining suspension periods for the Alert Monitor	203

Operating the Replication Alert Monitor	204
Starting monitors	205
Reinitializing monitors	205
Suspending and resuming a monitor	206
Ending a monitor suspension	206
Stopping monitors.	207
Reviewing Monitor program messages	207
Parameters of the Replication Alert Monitor	207
Default values of Replication Alert Monitor parameters	208
Descriptions of the Replication Alert Monitor parameters	208
Changing runtime parameters for the Replication Alert Monitor	211
Specifying how often the Replication Alert Monitor runs	211
Specifying notification criteria for selected alert conditions	212
Specifying notification criteria for operational errors	212
Specifying prune intervals for data from the Replication Alert Monitor	212

Chapter 16. Replication services (Windows). 215

Description of Windows services for replication	215
Creating a replication service	216
Starting a replication service	217
Stopping a replication service	217
Viewing a list of replication services.	217
Dropping a replication service.	218

Chapter 17. Scheduling SQL replication programs on various operating systems 219

Scheduling programs on Linux and UNIX operating systems	219
Scheduling programs on Windows operating systems	219
Scheduling programs on z/OS operating systems	220
Scheduling programs on the iSeries operating system	220

Chapter 18. How the SQL replication components communicate. 221

The Replication Center, ASNCLP, the Capture program or triggers, and the Apply program	221
The Capture program and the Apply program	222
The Capture triggers and the Apply program.	223
The administration tools and the Replication Alert Monitor	224
The Replication Alert Monitor, the Capture program, and the Apply program	225

Chapter 19. Viewing reports about the SQL replication programs 227

Checking the status of replication programs (Linux, UNIX, Windows, z/OS)	227
Reviewing historical data for trends	228

Reviewing Capture program messages	230
Examining Capture program throughput	230
Displaying latency of data processed by the Capture program	230
Reviewing Apply program messages	231
Examining Apply program throughput	232
Displaying the average length of time taken to replicate transactions	232
Checking the status of the Capture and Apply journal jobs (iSeries)	233
Monitoring the progress of the Capture program (iSeries)	233

Chapter 20. Customizing and running replication SQL scripts for SQL replication. 235

Chapter 21. Naming rules for SQL replication objects 237

Chapter 22. System commands for SQL replication (Linux, UNIX, Windows, z/OS) 239

asncap: Starting Capture.	239
asnccmd: Operating Capture	244
asnapply: Starting Apply	247
asnacmd: Operating Apply	253
asnanalyze: Operating the Analyzer	254
asnmon: Starting a Replication Alert Monitor.	257
asnmcmd: Working with a running Replication Alert Monitor	261
asnpwd: Creating and maintaining password files	264
ansrcrt: Creating a replication service	268
ansndrop: Dropping replication services	270
ansnlist: Listing replication services	271
asntdiff: Comparing data in source and target tables	272
asntrc: Operating the replication trace facility.	275
asntrep: Repairing differences between source and target tables	282

Chapter 23. System commands for SQL replication (iSeries) 287

ADDDPRREG: Adding a DPR registration (iSeries)	287
ADDDPRSUB: Adding a DPR subscription set (iSeries)	295
ADDDPRSUBM: Adding a DPR subscription-set member (iSeries)	310
ANZDPR: Operating the Analyzer (iSeries)	319
CHGDPRCAPA: Changing DPR Capture attributes (iSeries)	322
CRTDPRTBL: Creating the replication control tables (iSeries)	327
ENDDPRAPY: Stopping Apply (iSeries)	328
ENDDPRCAP: Stopping Capture (iSeries)	331
GRTDPRAUT: Authorizing users (iSeries)	333
INZDPRCAP: Reinitializing DPR Capture (iSeries)	341
OVRDPRCAPA: Overriding DPR Capture attributes (iSeries)	342

RMVDPRREG: Removing a DPR registration (iSeries)	347
RMVDPRSUB: Removing a DPR subscription set (iSeries)	348
RMVDPRSUBM: Removing a DPR subscription-set member (iSeries)	349
RVKDPRAUT: Revoking authority (iSeries)	351
STRDPRAPY: Starting Apply (iSeries)	353
STRDPRCAP: Starting Capture (iSeries)	359
WRKDPRTRC: Using the DPR trace facility (iSeries)	366

Chapter 24. SQL replication table structures 373

Tables at a glance	373
Tables at the Capture control server	380
IBMSNAP_CAPENQ table (Linux, UNIX, Windows, z/OS)	382
IBMSNAP_CAPMON table	382
IBMSNAP_CAPPARMS table	384
IBMSNAP_CAPSCHEMAS table	387
IBMSNAP_AUTHTKN table (iSeries)	388
IBMSNAP_CAPTRACE table	389
CCD table (non-DB2)	390
CD table	391
IBMSNAP_PARTITIONINFO table	392
IBMSNAP_PRUNCNTL table	393
IBMSNAP_PRUNE_LOCK table	395
IBMSNAP_PRUNE_SET table	396
IBMSNAP_REG_EXT (iSeries)	396
IBMSNAP_REGISTER table	398
IBMSNAP_REG_SYNCH table (non-DB2 relational)	404
IBMSNAP_RESTART table	405
IBMSNAP_SEQTABLE table (Informix)	407
IBMSNAP_SIGNAL table	407
IBMSNAP_UOW table	410
Tables at the Apply control server	412
ASN.IBMSNAP_APPENQ table	413
ASN.IBMSNAP_APPLY_JOB (iSeries)	413
ASN.IBMSNAP_APPPARMS table	414
ASN.IBMSNAP_APPLYTRACE table	417
ASN.IBMSNAP_APPLYTRAIL table	418
ASN.IBMSNAP_SUBS_COLS table	423
ASN.IBMSNAP_SUBS_EVENT table	424
ASN.IBMSNAP_SUBS_MEMBR table	425
ASN.IBMSNAP_SUBS_SET table	429
ASN.IBMSNAP_SUBS_STMTS table	434
List of tables at the Monitor control server	436
IBMSNAP_ALERTS table	437
IBMSNAP_CONDITIONS table	438
IBMSNAP_CONTACTGRP table	443
IBMSNAP_CONTACTS table	444
IBMSNAP_GROUPS table	445
IBMSNAP_MONENQ table	445
IBMSNAP_MONPARMS table	445
IBMSNAP_MONSERVERS table	447
IBMSNAP_MONTRACE table	448
IBMSNAP_MONTRAIL table	449
IBMSNAP_SUSPENDS table	451
IBMSNAP_TEMPLATES table	451

List of tables used at the target server	452
Base aggregate table	453
Change aggregate table	453
Consistent-change data (CCD) table	454
Point-in-time table.	456
Replica table	457
User copy table.	457

Chapter 25. UNICODE and ASCII encoding schemes for SQL replication (z/OS). 459

Rules for choosing an encoding scheme	459
Setting encoding schemes	459

Chapter 26. Starting the SQL replication programs from within an application (Linux, UNIX, Windows). . . 461

Chapter 27. How the Capture program processes journal entry types for SQL replication (iSeries). 463

Accessing information about IBM . . . 465

Contacting IBM	465
Accessible documentation	466
Providing comments on the documentation	466

Notices and trademarks 467

Notices	467
Trademarks	469

Index 471

Chapter 1. Planning for SQL replication

The following topics describes how to plan your replication environment:

Migration planning

Planning migration involves planning for issues that might arise while migrating from one version of replication to another.

If you are migrating from an existing replication environment, certain migration issues need to be considered. *WebSphere® Information Integration Migrating to Replication Version 9* describes how to migrate to Version 9 replication. To migrate to Version 9, your servers must first be at Version 8. *WebSphere Information Integration Migrating to SQL Replication Version 8* describes how to migrate 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. These documents are available online at from the WebSphere Information Integration support site for your product.

Memory planning

Memory planning involves planning for the amount of memory required by replication. 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. For more information, see the following topics.

Memory used by the Capture program

The Capture program uses memory when it reads the DB2 recovery log. It 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 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. 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 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 iSeries™

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 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 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 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

The Apply program uses memory when it fetches data. 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.

Storage planning

Storage planning is important for log impact for DB2 source servers, log impact for target servers, storage requirements of target tables and control tables, space requirements for diagnostic log files (Linux, UNIX, Windows, z/OS), space requirements for spill files for the Capture program, and space requirements for the spill files for the Apply program.

In addition to the storage required for DB2, you must ensure that storage is available for replication for the following topics. All of the sizes given in these topics 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 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.

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 percent 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.

iSeries:

- 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 RGZCTLBTL (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.

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.

iSeries: If the target operating system is iSeries, you also need to consider the log space (journal receivers space) of the target tables. Because journal receivers for target tables on iSeries 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:

$$\text{journal_receiver_volume} = \text{target_table_row_length} \times \text{journal_receiver_threshold}$$

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:

$$\text{recommended_CD_size} = (21 \text{ bytes} + \text{sum}(\text{length of all registered columns})) \times (\text{number of inserts, updates, and deletes to source table during the contingency period})$$

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.

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.
- **iSeries:** On iSeries, 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.

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).

Linux, UNIX, Windows, z/OS: The row size of the spill file 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.

iSeries: The row size of the spill file is a constant 32 KB.

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.

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 the diagnostic log 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.

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.

Conflict detection planning

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 codes¹ 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.

1. United States postal codes.

Non-DB2 relational source planning

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:

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 increase the response time for updating transactions. The impact is greatest for those transactions that heavily update application source tables that are to be replicated.

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.

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.

Coexistence of preexisting 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 preexisting 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.

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.

Code page translation planning

Replication components are database applications that rely on the DB2 databases on various operating systems to handle code page translation of data.

Replication components work with data using SQL SELECT, INSERT, UPDATE, and DELETE statements. For more information, see the following topics:

Replication for 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 Information Center 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.

Configure 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 or Apply programs on Linux, UNIX or Windows operating systems, some configuration steps may be necessary.

On Linux, UNIX, and Windows, the Capture program must run in the same code page as the database from which it is capturing the data. If the Capture program is not run in the same code page, you must set a DB2 environment variable or registry variable called DB2CODEPAGE so that Capture uses the same code page as the database.

When running the Apply program on Linux, UNIX, or Windows, if any source table is in UNICODE, the Apply application code should be in UNICODE. If the data in the source table is in ASCII, the application code page can be in ASCII or UNICODE. You can also set the DB2CODEPAGE variable for the Apply program.

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 or Apply programs 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. Otherwise, data might not be translated correctly. The value that you use for the DB2CODEPAGE variable must be the same as what you specify on your CREATE DATABASE statement. See the DB2 Information Center for details 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, DB2 on iSeries 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 iSeries 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 /sqlib/msg/en_US subdirectory. If Capture cannot find its message library, all messages written to the IBMSNAP_CAPTRACE table are ASN0000S.

Replication planning for DB2 UDB for z/OS

SQL replication for z/OS 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 for Linux, UNIX, Windows or DB2 UDB for 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 for Linux, UNIX, Windows or DB2 UDB for iSeries.

Performance tuning

Performance tuning involves tuning your replication environment for optimal performance.

WebSphere Information Integration Tuning for SQL Replication Performance describes how to tune the major components of a DB2 replication environment for maximum performance. This document is available online the WebSphere Information Integration support site for your product.

Chapter 2. Setting up user IDs and passwords for SQL replication

To use the SQL replication programs, you need to set up user IDs and passwords for accessing DB2 servers on local and remote systems.

The following topics explain the authorities and privileges that you need, and how you can store necessary user IDs and passwords in an encrypted password file that the SQL replication programs can share.

Authorization requirements for administration

To set up replication, you run generated SQL to create objects, and bind plans and create SQL packages (iSeries). Authorities required for these tasks vary by operating system.

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.

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 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 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 iSeries

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 iSeries systems, you should use the same user ID for all servers.

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.

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.
- SELECT privilege for the DB2 catalog (SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, and SYSIBM.SYSPLAN).

- 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_09_01/bin, must be in your PATH.

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 directory specified by the **capture_path** parameter. The Capture program creates diagnostic files in this directory.

iSeries

Use the Grant DPR Authority (GRTPRAUT) command to authorize a user to run the Capture program on a local system. If you are replicating between only iSeries systems, you should use the same user ID for all servers. If the GRTPRAUT command is not installed on a machine, you must use the Grant Object Authority (GRTOBJAUT) command.

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 different user IDs at each server in your replication environment. You can use the replication administrator user ID to run the Apply program, but this is not a requirement.

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.
- SELECT authority for the DB2 catalog (SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, and SYSIBM.SYSPLAN).

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_09_01/bin, must be in your PATH.

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 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 federated database and in the non-DB2 relational database to access the source tables through nicknames, which are defined on the federated database.

non-DB2 databases

If your control tables are on non-DB2 databases, 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 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 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 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 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 federated database and, through user mappings, the privilege to READ from the actual source table on the non-DB2 server.

iSeries

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 iSeries 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 database, 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.

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

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.

If your replication environment is not distributed across servers, you do not 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. For information about password requirements for the Replication Center, see the online help.

Chapter 3. Configuring servers for SQL replication

Before you can replicate data, you must create and configure your servers and ensure that they can connect to each other.

The following topics explain how to set up and configure your environment. For more detail about configuring servers on z/OS, see *WebSphere Information Integration Replication Installation and Customization Guide for z/OS*.

Connectivity requirements for SQL 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.

The following topics provide detail about connectivity requirements.

Connecting to iSeries servers from Windows

You can administer replication on iSeries servers by connecting from a Windows workstation.

Before you begin

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

Procedure

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 iSeries database in DB2 for Windows:

- a. From a Windows command prompt, enter db2clp. 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
```

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.

For more detail on configuring a federated environment, see the DB2 Information Center or the *WebSphere Information Integration Federated Systems Guide*.

Creating control tables for SQL replication

The replication programs use control tables to store information about registered tables, subscription sets, operational parameters, and user preferences. You create control tables before defining your sources and targets for replication.

The following topics describe creating control tables for SQL replication.

Creating control tables for SQL replication

You can use the Replication Center or the ASNCLP command-line program to create control tables for the Capture and Apply programs.

Restrictions

- The Replication Center or ASNCLP must be able to connect to the server where you want to create the control tables.
- In a multiple database partition environment, all of the table spaces that are used by the control tables must be on the catalog partition. If you use an existing table space, the table space must be non-partitioned and it must be on the catalog partition.

About this task

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.

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.

Both replication administration tools allow you to create a profile for to identify the defaults to be used when you create control tables for a given operating system or environment. 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. 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.

Procedure

To create control tables, use one of the following methods:

Method	Description
ASNCLP command-line program	Use the CREATE CONTROL TABLES FOR command to create a new set of Capture or Apply control tables. For example, the following commands set the environment and create Capture control tables: <pre>SET SERVER CAPTURE TO DB SAMPLE SET OUTPUT CAPTURE SCRIPT "capctrl.sql"; SET LOG "capctrl.err"; SET RUN SCRIPT LATER; CREATE CONTROL TABLES FOR CAPTURE SERVER IN UW UOW TSUOW100 OTHERS TSASN100;</pre>
Replication Center	Use either the Create Control Tables or Create Control Tables - Quick windows for Capture and Apply. To open the windows, right-click a Capture Control Servers or Apply Control Servers folder in the object tree and click one of the following options: <ul style="list-style-type: none">• Create Capture Control Tables• Create Capture Control Tables → Quick• Create Apply Control Tables• Create Apply Control Tables → Quick

Creating control tables (iSeries)

Replication control tables are created automatically when you install DB2 DataPropagator for iSeries. You can also use a command to create control tables.

About this task

During the installation, control tables are created in the DataPropagator default schema (ASN), if they do not already exist. You can create additional sets of control tables if your control tables are accidentally deleted or corrupted. For Capture, you can create the new set of control tables with a different schema. You can create a maximum of 25 schemas.

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.

Procedure

To create control tables on iSeries, use the Create DPR Tables (CRTDPRTBL) command.

Restriction: Use only the CRTDPRTBL command to create control tables on iSeries. The ASNCLP command-line program and Replication Center do not support the creation of control tables for iSeries.

Creating control tables for non-DB2 relational sources

If you want to use a non-DB2 database such as Informix as a replication source, you can use the Replication Center or ASNCLP command-line program to create control tables.

About this task

For these types of sources, the Replication Center creates the following Capture control tables in the non-DB2 relational database:

- IBMSNAP_PRUNCNTL
- IBMSNAP_PRUNE_SET
- IBMSNAP_REG_SYNCH
- IBMSNAP_REGISTER
- IBMSNAP_SEQTABLE on Informix only
- IBMSNAP_SIGNAL

Nicknames are created in a federated database for all but IBMSNAP_SEQTABLE. (This table is used only by the Informix triggers. The Apply program does not use it.) Triggers are created automatically on the IBMSNAP_SIGNAL table and the IBMSNAP_REG_SYNCH table.

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.

About this task

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.

z/OS: You can use multiple Capture schemas if 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.

iSeries: 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.

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 partition.

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. In the Replication Center, 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.

Setting up the replication programs

Before you can replicate, you need to set up the Capture program, Apply program, and other replication programs for the servers in your environment.

The following topics describe required setup for the replication programs.

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

To set up the replication programs you need to set environment variables, prepare the database for the Capture program, and optionally bind packages.

The following topics describe these tasks.

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:

Operating system	Command
Linux and UNIX	<code>export DB2INSTANCE=db2_instance_name</code>
Windows	<code>SET DB2INSTANCE=db2_instance_name</code>

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.
Note: 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.
3. **Optional:** Set environment variable DB2DBDFT to the source server.
4. **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.

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

To prepare the DB2 database to run the Capture program, you enable archival logging. You can also set other database configuration parameters.

Procedure

To prepare the DB2 database to run the Capture program:

1. Connect to the Capture control server database by entering the following command:
`db2 connect to database`

Where *database* is the Capture control server database.

2. Enable archival logging and prepare the Capture control server database for roll-forward recovery by issuing the update database configuration command (logretain recovery) and the backup database command. 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.
3. 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 value of the Capture **memory_limit** parameter.
 - 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 Capture program is bound automatically on Linux, UNIX, and Windows during execution. You can bind packages manually if you want to specify bind options, schedule binding, or check that all bind processes completed successfully.

Procedure

To bind the Capture program packages:

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

```
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.

Linux and UNIX:

```
db2homedir/sqllib/bndWhere db2homedir is the DB2 instance home directory.
```

Windows:

```
drive:...\sqllib\bndWhere drive: is the drive where DB2 is installed.
```

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)

The Apply program is bound automatically on Linux, UNIX, and Windows during execution. You can bind packages manually if you want to specify bind options, schedule binding, or check that all bind processes completed successfully.

Procedure

To bind the Apply program packages:

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

Linux and UNIX:

```
db2homedir/sqllib/bnd
```

Where *db2homedir* is the DB2 instance home directory.

Windows:

```
drive:\...\sqllib\bnd
```

Where *drive:* is the drive where DB2 is installed.

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 the following command:

```
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
```

3. 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*.

Creating SQL packages to use with remote systems (iSeries)

You need to create packages using the CRTSQLPKG command in some cases on iSeries.

About this task

Use this command to create packages in the following cases:

- When using remote journaling. Run the CRTSQLPKG command on the system where the Capture program is running and 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 and use the following guidelines:
 - 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 iSeries systems or between an iSeries system and some other operating system (such as Linux, UNIX or Windows).

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

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

The following topics provide more detail on creating SQL packages:

Creating SQL packages for the Apply program (iSeries)

You must create SQL packages for the Apply program on iSeries so it can interact with all the remote servers to which it needs to connect.

Procedure

To create SQL packages for the Apply program:

Run this command on the system where Apply is running to enable it to connect to a remote system:

```
CRTSQPKG 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 (iSeries)

You must create SQL packages for the Replication Analyzer on iSeries so it can interact with the servers that you are analyzing, such as the Capture control server or the target server.

Procedure

To create SQL packages for the Replication Analyzer:

Run this command on the system where the Replication Analyzer is running:

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

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

Granting privileges to the SQL packages (iSeries)

After you create SQL packages on iSeries, you must grant *EXECUTE privileges to all users who will be subscribing to files registered on the source database.

Procedure

To grant privileges for SQL packages:

Log on to the iSeries system where the source database resides and use one of the following methods:

Method	Description
GRTOBJAUT command	Use the Grant Object Authority (GRTOBJAUT) command, for example: GRTOBJAUT OBJ(<i>ASN/package_name</i>) OBJTYPE(*SQLPKG) USER(<i>subscriber_name</i>) AUT(*OBJOPR *EXECUTE)
GRANT SQL statement	Use SQL to connect to the source database and run the GRANT SQL statement: CONNECT TO <i>data_server_RDB_name</i> GRANT EXECUTE ON PACKAGE <i>ASN/package_name</i> TO <i>subscriber_name</i>

Method	Description
GRTDPRAUT command	Use the GRTDPRAUT command, if it is installed on the local system.

Setting up the replication programs (z/OS)

You must set up and customize the replication programs when you install SQL replication on z/OS.

See the instructions in the *WebSphere Information Integration Replication Installation and Customization Guide 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 or 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 parameter to any of the warm start modes on the Start Capture window.

Setting up journals (iSeries)

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 iSeries, the table must be journaled for both before-images and after-images.

The following topics describe the journal setup required for replication.

Setting up journals for source tables (iSeries)

To set up journaling for a source table, you create a journal receiver, create a journal, and then start journaling.

Before you begin

You must have authority to create journals and journal receivers for the source tables to be defined.

Restrictions

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

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, as in the following example:

```
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 *iSeries Programming: Performance Tools Guide* for more information.

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

```
STRJRNP 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 (iSeries)

The Capture program uses the Receive Journal Entry (RCVJRNE) command to receive journals.

The following topics describe how to manage journals and journal receivers.

Specifying system management of journal receivers (iSeries):

You should let the iSeries system manage the changing of journal receivers. This is called *system change journal management*.

Restrictions

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.

Procedure

To specify system management of journal receivers, 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.

Changing definitions of work management objects (iSeries):

You can alter the default definitions for the three types of work management objects that are created during installation of DB2 DataPropagator for iSeries, or provide your own definitions.

About this task

The installation program creates an SQL journal, an SQL journal receiver for this library, and work management objects.

Table 1 lists the objects that are created.

Table 1. Work management objects

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

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 Guide* (SC41-5306) for more information about changing these definitions.

Specifying user management of journal receivers (iSeries):

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.

About this task

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 *iSeries Backup and Recovery*.

Delete journal receiver exit routine (iSeries):

The *delete journal receiver* exit routine (DLTJRNRCV) helps ensure that journal receivers are not deleted if the Capture program has not processed all the entries they contain.

When you install DB2 DataPropagator for iSeries, this exit routine is registered automatically. It 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.

Attention: 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 DLTLOPT(*IGNEXITPGM) to override the exit routine.

Chapter 4. Registering tables and views as SQL replication sources

With SQL 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 section 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 SQL replication captures data and changes from the source.

For SQL 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 section contains the following topics.

Registering DB2 tables as sources

When you register a DB2 table as a replication source, you specify the source server, source table name, and the Capture schema. A CD (change-data) table is created for you.

Before you begin

- For all DB2 sources except for iSeries, the source table DDL requires the `DATA CAPTURE CHANGES` option. Do not remove this option from your source.
- Capture control tables must already exist on the Capture control server that will process the table that you want to register as a source.

Restrictions

iSeries

- 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.

About this task

SQL replication supports the following types of DB2 tables as sources:

z/OS

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

iSeries

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

Linux, UNIX, and Windows

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

You can register the same table multiple times using different Capture schemas.

Procedure

To register a DB2 table, use one of the following methods:

Method	Description
ASNCLP command-line program	Use the CREATE REGISTRATION command to register a source table, view, or nickname. For example, the following commands set the environment and register the STAFF table in the DB2 SAMPLE database. <pre>SET SERVER CAPTURE TO DB SAMPLE; SET OUTPUT CAPTURE SCRIPT "register.sql"; SET LOG "register.err"; SET RUN SCRIPT LATER; CREATE REGISTRATION (DB2ADMIN.STAFF) DIFFERENTIALREFRESH STAGE CDSTAFF;</pre>
Replication Center	Use the Register Tables window. In the object tree, expand your chosen Capture schema, right click the Registered Tables folder, and click Register Tables . 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.
ADDDPRREG system command (iSeries)	Use the ADDDPRREG command to register a table on iSeries. For example, 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: <pre>ADDDPRREG SRCTBL(HR/EMPLOYEE) CAPCTLLIB(BSN) CDLIB(HRCDLIB) CDNAME(CDEMPLOYEE)</pre>

When you register a table as a source, 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 rolls back. 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 IBMSNAP_REGISTER table so that the Apply program can read from this source table correctly.

Registering non-DB2 relational tables as sources

When you register a non-DB2 relational table, you specify the nickname of the source table that you want to register. A CCD (consistent-change data) table is created for you.

Before you begin

Capture control tables must already exist on the Capture control server that will process this source.

Restrictions

- If you are using a single federated 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 schemas can be the same. You can register a non-DB2 relational table under only one Capture schema.
- You cannot register LOB columns in non-DB2 relational tables. If you register a table that includes this data type, you must register a column subset.

About this task

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.

Procedure

To register a non-DB2 relational table, use one of the following methods:

Method	Description
ASNCLP command-line program	<p>Use the CREATE REGISTRATION command to register a source table, view, or nickname. For example, the following commands set the environment and create a registration with the following characteristics:</p> <ul style="list-style-type: none"> • Non-IBM server that contains the Oracle database V9ORA • Federated server FEDORADB • CCD table in the Oracle database undjr09.ccdtest • CCD nickname in the federated server repldba.ccdtestnk • Source nickname that is being registered repldba.testnk • All columns in repldba.testnk are registered with after images <pre>SET SERVER CAPTURE TO DB FEDORADB NONIBM SERVER V9ORA ID repldba PASSWORD "passwd"; SET OUTPUT CAPTURE SCRIPT "ora_reg.sql"; SET CAPTURE SCHEMA SOURCE ASNORA; SET LOG "orareg.out"; CREATE REGISTRATION (repldba.testnk) DIFFERENTIALREFRESH STAGE repldba.ccdtestnk CONDENSED OFF NONIBM undjr09.ccdtest COLS ALL IMAGE AFTER;</pre> <p>The CONDENSED OFF option is required for federated sources.</p>
Replication Center	<p>Use the Register Nicknames window. From the object tree, expand the non-DB2 relational database that contains the nicknames that you want to register. Right-click the Registered Nicknames folder and select Register Nicknames.</p> <p>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.</p>

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.

Registration options for source tables

SQL replication provides many options when you register a table as a replication source. These options are part of the larger task of registering a table.

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).

The following topics detail options for registering tables:

Registering a subset of columns (vertical subsetting)

You can register a subset of the source table columns for replication, for example if you do not want all of the columns available for targets to subscribe to or if target tables do not support all data types that are defined for the source table.

By default, all columns are registered. To register a subset of the columns, select only those columns that you want to make available for replication to a 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 source columns only if you are sure that you will never want to replicate the unregistered columns. If you later want to replicate columns that you didn't register, you must 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 and use the Apply program to subset which columns are replicated to targets.

Change-capture replication and full-refresh copying

By default, only changes that occurred at the source table since the last replication cycle are replicated (change-capture replication). You can also replicate all data in the source table during each cycle (full-refresh-only replication).

Change-capture replication

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.

Tip: 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.

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.

Small 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.

Large 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.

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.

After-image columns and before-image columns

When you register a source for change-capture replication, by default only the changed (after-image) value in a column is captured. You can also choose to capture the previous (before-image) value.

Linux, UNIX, Windows, and z/OS

You can select whether to capture before-image values for individual columns in a table.

iSeries

You can select whether to capture before images for all or none of the columns in a table. You cannot select this option for each individual column.

Sybase or Microsoft® SQL Server

A 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.

Restriction: You cannot include before-image values in the CD table for columns with LOB data types.

The sections below discuss when you should choose each option.

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 do not 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.

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:

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.

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

Important for z/OS: For tables in DB2 for z/OS, you can use 18-character column names, but replication 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.

Important for Linux, UNIX, Windows, and iSeries: 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, replication truncates the additional characters from the right by default, unless you have set your profile to truncate from the left. Because 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.

The following list describes cases in which you might want to capture before-image values:

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, SQL replication requires special handling to ensure that the correct rows at the target table are updated. To ensure that SQL 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 CD (or CCD) table.

Although you register these before-image values when you register the source table or view, 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.

Before-image prefix

If you capture after-image and before-image columns, the after-image column takes the name of the column at the source table, and the before-image column takes the name of the column at the source table with a one-character prefix.

The default before-image prefix assigned by the ASNCLP command-line program and Replication Center is X. The default for the iSeries system commands is @.

You can change the default prefix. The combination of the before-image prefix and the CD (or CCD) column name cannot be the same as a current or potential column name in the CD (or CCD) table.

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.

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

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

When the Capture program detects certain problems while processing registrations, by default it stops. You can choose to let the program keep running.

The following list provides detail to help you choose the best option for your environment.

Stop Capture on error

With this option, the Capture program writes an error message in the IBMSNAP_CAPTRACE table and terminates.

The Capture program stops 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.

The Capture program does not stop when certain non-fatal errors occur, for example:

- SQLCODES indicate invalid length of data.
- **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 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-iSeries) 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.

Options for how the Capture program stores updates

By default updates to the source table are stored in a single row in the CD table. In some cases you should instruct the Capture program or triggers to capture updates as DELETE and INSERT pairs that are stored in two rows.

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.

Example: 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.

Preventing the recapture of changes (update-anywhere replication)

For update-anywhere replication, you can use the recapture option to control whether changes that are replicated from one site are recaptured at the second site for replication to additional sites.

Restriction: Tables from non-DB2 relational databases cannot participate in update-anywhere. 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, SQL replication assumes that it will be the master table in your configuration.

During registration, you set the recapture 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.

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 is not capturing the same changes again for each replica.

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

Masters with only one replica

If you plan to have only one replica in your update-anywhere configuration, create your registration so that changes are not recaptured at either the master table or the replica table.

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.

Multiple replicas that are mutually exclusive partitions of the master

For multiple replicas that are mutually exclusive partitions of the master, create your registration so that changes are not recaptured at either the master table or the replica tables.

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.

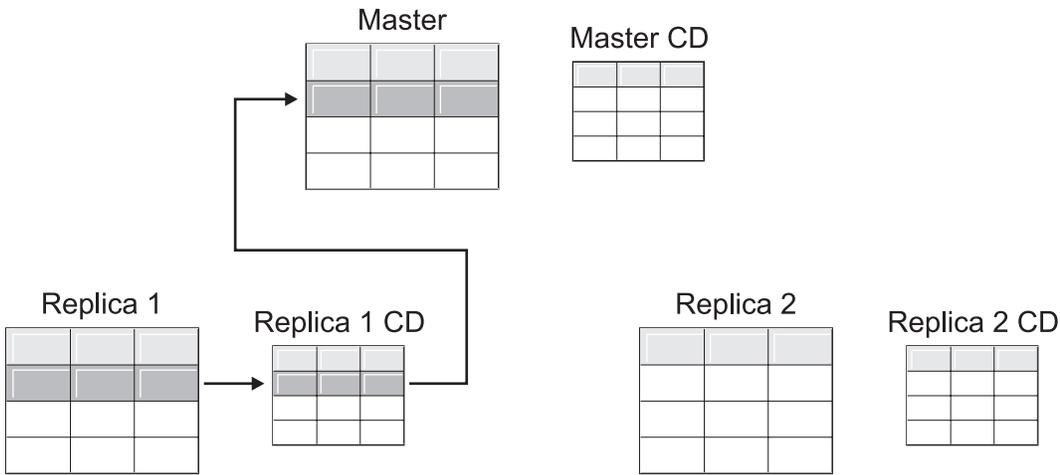


Figure 1. Recapture option for replicas that are mutually exclusive partitions of the master. When you have multiple replicas that do not subscribe to the same data in the master, you do not need to use the recapture option for any of the tables.

Masters that replicate changes to multiple replicas

For masters that replicate changes to multiple replicas, create your registration so that changes are recaptured at the master table but not recaptured at the replica tables.

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.

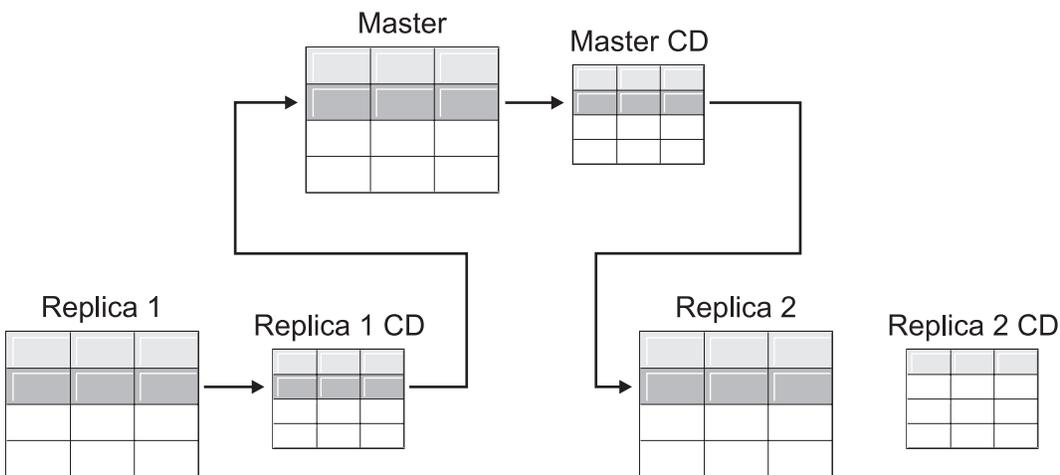


Figure 2. Recapture option for masters that replicate changes to multiple replicas. When you have multiple replicas that subscribe to the same data in the master, you can use the recapture option at the master so that changes that occur at one replica are recaptured at the master and forwarded to the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

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

For replicas that replicate changes to other replicas (multi-tier), create your registration so that changes are not recaptured at the master table but are recaptured at the replica tables.

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).

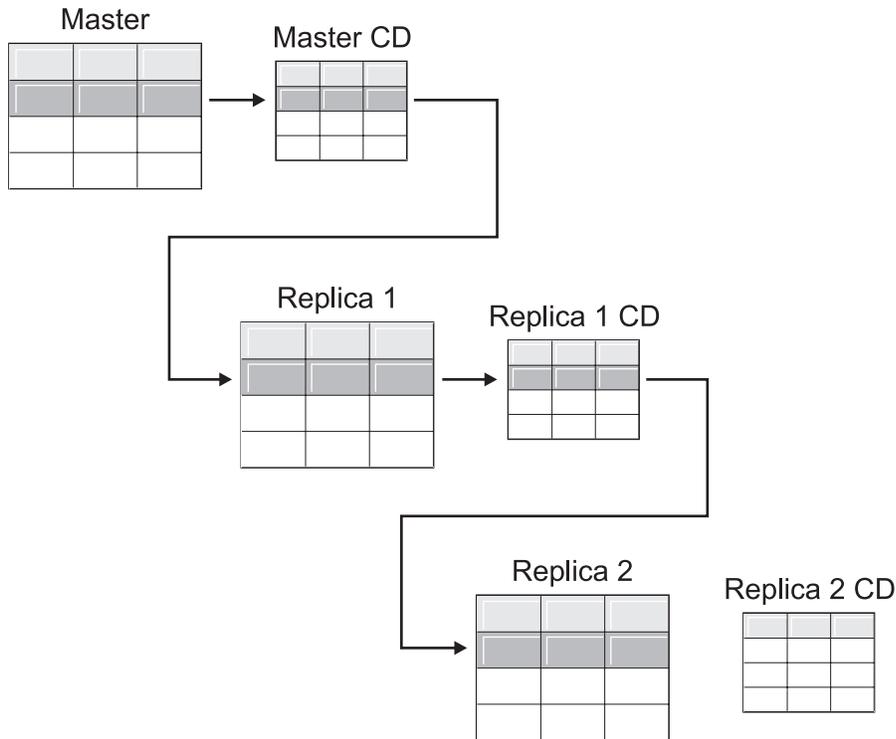


Figure 3. Recapture option at tier 2 allows changes at tier 1 to be replicated down to tier 3. When you have a replica table that acts as a middle tier in a multi-tier configuration, you can use the recapture option at the replica so that changes that occur at the master are recaptured at the replica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

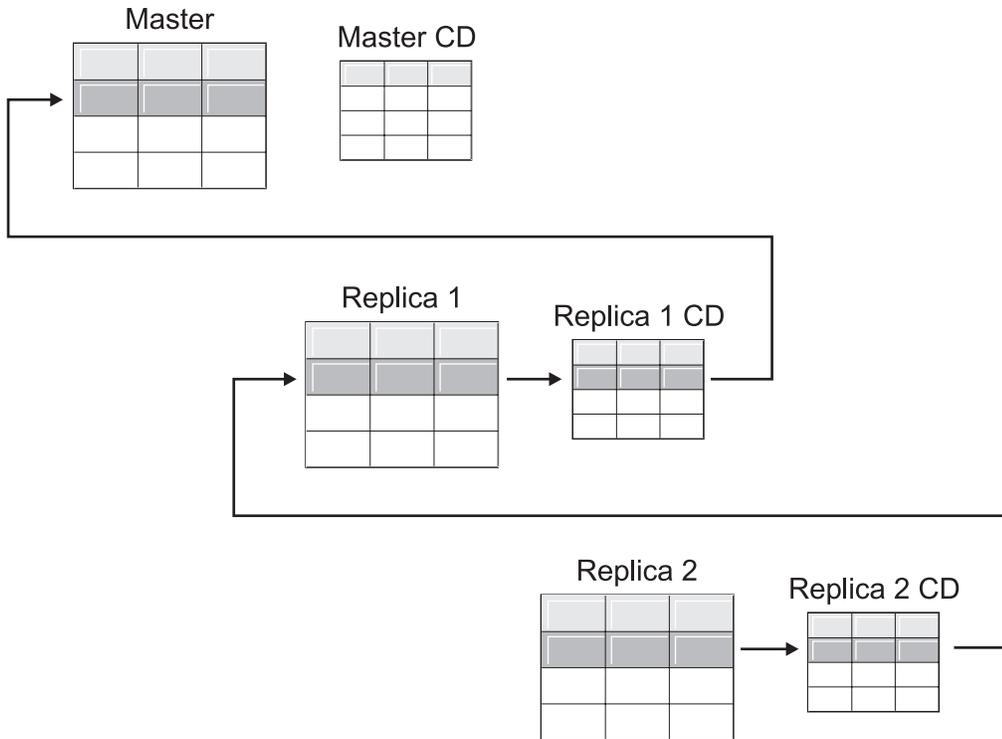


Figure 4. Recapture option at tier 2 allows changes at tier 3 to be replicated up to tier 1. When you have a replica table that acts as a middle tier in a multi-tier configuration, you can use the recapture option at the replica so that changes that occur at the replica in the subsequent tier are recaptured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

Options for conflict detection (update-anywhere replication)

In update-anywhere configurations, conflicts can sometimes occur between the master and its replicas. When you register a source, you can select among three levels of conflict detection: none, standard, and enhanced.

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.

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 LOB columns, you must specify None for the conflict-detection level.

Based on your tolerance for lost or rejected transactions and performance requirements, you can decide which type of detection to use:

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.

Registering tables that use remote journaling (iSeries)

When registering iSeries tables that use remote journaling, you can specify 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 iSeries database server that is separate from the iSeries server that the source table is on.

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

Recommendation: Whenever you are replicating data from one iSeries table to another iSeries 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 iSeries system as the replication target. SQL replication allows you to register remote journals as sources even if the registration does not reside on the same iSeries 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 iSeries 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 *iSeries Remote Journal Function for High Availability and Data Replication*, SG24-5189.

Using relative record numbers (RRN) instead of primary keys (iSeries)

If you are registering an iSeries 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 of their underlying tables, particularly the option of change-capture or full-refresh replication.

The following topics describe how registered views behave in various scenarios.

Views over a single table

You can register a view over a single table if the underlying table is registered for replication. The view inherits the type of replication from the underlying table.

Full refresh only

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.

Change capture

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 CD table of the underlying 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, at least one of the underlying tables in the join must be registered. You can also have inner-joins of CCD tables that are registered as sources.

When you register a join as a replication source, SQL replication adds multiple rows in the 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;
```

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

Table 2. 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.

Avoiding double deletes

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

When you register a view as a source for replication, the view inherits the registration options of the source table on which the view is based.

Before you begin

- Capture control tables must already exist on the Capture control server that will process the view that you want to register as a source.
- 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.

Restrictions

- You cannot register views of non-DB2 relational tables.
- You cannot register a view that is over another view.
- All CCD tables that have views defined over them must be complete and condensed to be registered as a replication source.
- **iSeries:** 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.

Procedure

Use one of the following methods to register a view:

Method	Description
ASNCLP command-line program	Use the CREATE REGISTRATION command and specify the view name for the <i>objowner</i> (object owner) and <i>objname</i> (object name). For views, the command decides whether the source can be registered as differential or full refresh.
Replication Center	Use the Register Views window. Expand the Capture schema under which you want to register views. Right-click the Registered Views folder and click Register Views .
ADDDPRREG system command (iSeries)	Use the ADDDPRREG command to register a view on iSeries.

Maintaining CCD tables as sources (IMS)

If you have externally populated CCD tables that are 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.

Procedure

To maintain a CCD table that is populated by an external tool:

Update three columns in the IBMSNAP_REGISTER table (CCD_OLD_SYNCHPOINT, SYNCHPOINT, and SYNCHTIME) for each of the following types of events:

Event	Required updates
Initial full refresh or load of the CCD table	<ul style="list-style-type: none">• 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.
Any update to the CCD table after the full refresh or load	<ul style="list-style-type: none">• 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.
Any subsequent full refresh or load of the CCD table	<ul style="list-style-type: none">• 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_SYNCHPOINT value is larger than the most recently applied SYNCHPOINT value.

Chapter 5. Subscribing to sources for SQL replication

After you register tables or views as replication sources, you can define a subscription for your target tables or views so that they receive the initial source data and subsequent changes.

The administration tasks described in this section 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.

The following topics provide details on subscribing to sources.

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.

SQL 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 iSeries 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 topics discuss the trade-offs in grouping subscription sets per Apply program and 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 `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.

Before you begin

- Create the Apply control tables in the Apply control server for the subscription set.
- Before you add subscription-set members to subscription sets, register the tables or views that you want to use as sources. You should also consider how you want to group your sets.

About this task

When you create a subscription set, you specify the source and target servers, which Capture 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. You can create an empty 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.

Procedure

To create a subscription set, use one of the following methods:

Method	Description
ASNCLP command-line program	<p>Use the CREATE SUBSCRIPTION SET command. This command can create only empty subscription sets, whereas the Replication Center allows you to add members to the set while creating it.</p> <p>The following commands set the environment and create a subscription set named SET00 with an Apply qualifier of AQ00.</p> <pre>SET SERVER CAPTURE TO DB SAMPLE; SET SERVER CONTROL TO DB TARGET; SET OUTPUT CAPTURE SCRIPT "capsubset.sql" CONTROLSRIPT "appsubset.sql"; SET LOG "subset.err"; SET RUN SCRIPT LATER; CREATE SUBSCRIPTION SET SETNAME SET00 APPLYQUAL AQ00 ACTIVATE YES TIMING INTERVAL 1 START DATE "2006-10-22" TIME "09:00:00.000000";</pre>
Replication Center	<p>Use the Create Subscription Set notebook. To open the notebook, expand the Apply control server where the set will be defined, right click the Subscription Sets folder and click Create.</p>
ADDDPRSUB system command (iSeries)	<p>Use the Add DPR subscription set (ADDDPRSUB) command to create a subscription set with either one member or no members.</p> <p>For example, to create a subscription set named SETHR under the AQHR Apply qualifier:</p> <pre>ADDDPRSUB APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/EMPLOYEE) TGTTBL(TGTLIB/TGTEMPL)</pre> <p>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.</p>

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, ASNCLP, or the Apply program so that the administration tools 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.

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, ASNCLP, or the Apply program so that the administration tools 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, ASNCLP, or the Apply program so that the administration tools 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.

Processing options for subscription sets

When you create a subscription set, you define options for how the Apply program processes the set.

The following topics help you to decide which settings to select based on your replication needs.

Specifying whether the subscription set is active

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

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.

This option is useful in several situations:

- 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_SYNC_MINUTES` column of the `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_SYNC_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_SYNC_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_SYNC_MINUTES` to 5 minutes.
- If `SLEEP_MINUTES` is set to 30 minutes (or more) for a given subscription set, set `MAX_SYNC_MINUTES` to 60 minutes.
- For `SLEEP_MINUTES` between 5 and 30 minutes, set `MAX_SYNC_MINUTES` equal to `SLEEP_MINUTES`.

Monitor your replication environment and adjust the `MAX_SYNC_MINUTES` as needed. Ensure that the numeric value for `MAX_SYNC_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 shows how the changed data is broken down into subsets of changes.

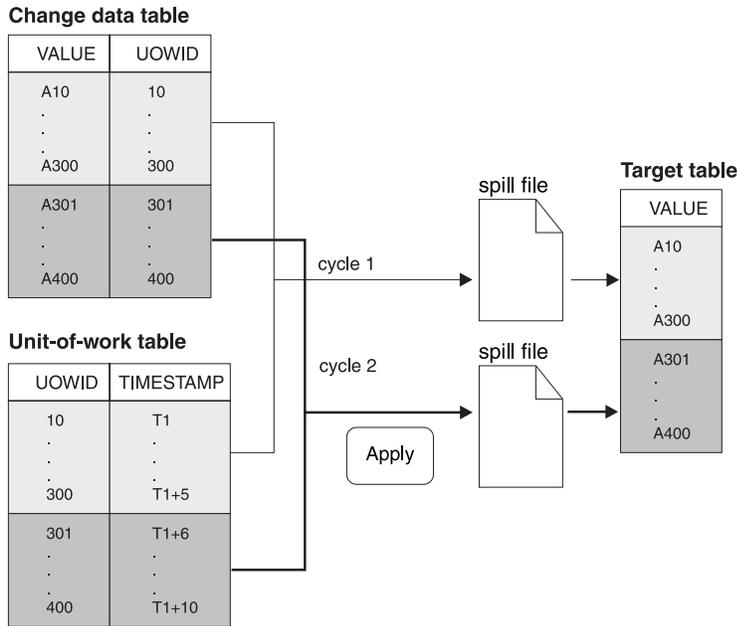


Figure 5. Data blocking. You can reduce the amount of network traffic by specifying a data-blocking value.

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.

Load options for target tables with referential integrity

In some cases you might want to postpone adding referential integrity constraints between target tables until after these tables are loaded with source data.

You decide how targets will be loaded when you set startup parameters for the Apply program. Consider these alternatives for creating referential integrity relationships between the target tables:

Before target tables are loaded

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 to bypass referential constraint checking during the load. If you do not use the LOADX option, the inserts into the target table could fail. A full refresh will typically be much faster using the LOADX startup option.

After the load completes and Apply has completed one cycle of applying changes to the targets

With this option, changes can 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. During the initial population of the target tables, the targets might be out of synch with each other regarding their referential integrity relationships. As the tables are 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.

Specifying how the Apply program replicates changes for subscription set members

When a subscription set has change-capture replication, you can decide whether the Apply program commits changes to the target table or view once for each subscription-set member or after applying a number of transactions.

After target tables are initially loaded, the Apply program starts to read the CD (or CCD) tables and collects the changes into spill files. The program then applies changes in one of two ways:

Table mode

The Apply program commits changes once for each subscription-set member.

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 commits changes after applying a number of transactions that you specify. Use transaction-mode processing when you have referential integrity constraints on target tables in the subscription set.

In this mode, the Apply program opens all of the spill files at once and processes the changes at the same time. Changes are applied in the order in which they 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.

Transaction-mode processing only changes the Apply program's behavior for sets with user-copy and point-in-time target tables. Also, note the following limitations:

- Sets containing CCD tables are always processed in table mode.
- Sets containing replica tables are always processed in transaction mode.

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.

You can also use a mixture of table-mode and transaction-mode processing, depending on the target-table types in the subscription set.

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.

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 syntax.

Options for scheduling replication of subscription sets

You can specify how often the Apply program processes a subscription set to control the currency of data in your target tables. You can use time-based scheduling, event-based scheduling, or a combination of these options.

For example, you can set an interval of one day between apply cycles, and also specify an event that triggers the 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, the Apply program might not be able to process a subscription set until it finishes applying data for all sets in the prior interval or for the prior event. In this case, you might not get the expected replication latency, but you won't lose any data.

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 specify an event name when you define the subscription set. You must also populate the IBMSNAP_SUBS_EVENT table with a timestamp for the event name. When the Apply program detects the event, it begins replication.

The IBMSNAP_SUBS_EVENT table has four columns, as shown in Table 3.

Table 3. Example of data stored in the IBMSNAP_SUBS_EVENT 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	

The EVENT_NAME column stores 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 3, 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 this time, and defers all subsequent updates.

You or your applications must post events to the 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.

Scheduling the subscription set

Define subscription-set timing information after you map sources to targets (or create an empty subscription set).

After you map sources to targets (or create an empty subscription set), 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. 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.

Creating subscription-set members

Within a subscription set, you can add source-to-target mappings for the Apply program to process as a group. These source-to-target mappings are called subscription-set members.

Before you begin

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. You should also create a subscription set and plan for how many members you want to add in a set.

Restrictions

- SQL 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 columns in the view 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.
- **Windows, Linux, UNIX, z/OS:** You can define a maximum of 200 members for each subscription set.
- **iSeries:** You can define a maximum of 78 members for each subscription set.

About this task

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.

Procedure

To add a subscription-set member, use one of the following methods:

Method	Description
ASNCLP command-line program	<p>Use the CREATE MEMBER command to add a subscription-set member to an existing subscription set. For example, the following commands:</p> <ul style="list-style-type: none"> • Set the environment. • Create a profile, TBSPROFILE, to store options for the tablespace that is used by the target table. • Specify the SET00 subscription set, AQ00 Apply qualifier, and the STAFF source table. • Specify that a new target table, TRGSTAFF, is created as a user copy with all columns registered. <pre> SET SERVER CAPTURE TO DB SAMPLE; SET SERVER CONTROL TO DB TARGET; SET SERVER TARGET TO DB TARGET; SET OUTPUT CAPTURE SCRIPT "capmember.sql" CONTROLSCRIPT "appmember.sql" SET LOG "member.err"; SET RUN SCRIPT LATER; SET PROFILE TBSPROFILE FOR OBJECT TARGET TABLESPACE OPTIONS UW USING FILE "/tmp/db/ts/TSTRG.TS" SIZE 700 PAGES; CREATE MEMBER IN SETNAME SET00 APPLYQUAL AQ00 ACTIVATE YES SOURCE STAFF TARGET NAME TRGSTAFF DEFINITION IN TSTRG00 CREATE USING PROFILE TBSPROFILE TYPE USERCOPY COLSALL REGISTERED; </pre>
Replication Center	<p>Use one of the following notebooks:</p> <ul style="list-style-type: none"> • Create Subscription Set. Use this notebook when you create the subscription set. Expand the Apply control server where the set will be defined, right click the Subscription Sets folder and click Create. • 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. Right-click the subscription set and select Properties. • Add Members to Subscription Sets. Use this notebook to add one member to multiple subscription sets. Each member must use the same source. Right-click the subscription sets to which you want to add a member and select Add Member.
ADDDPRSUBM command (iSeries)	<p>Use the ADDDPRSUBM command to add a member to a subscription set. For example, to add a subscription-set member to a subscription set named SETHR under the AQHR Apply qualifier:</p> <pre> ADDDPRSUBM APYQUAL(AQHR) SETNAME(SETHR) SRCTBL(HR/YTDTAX) TGTTBL(TGTHR/TGTTAX) </pre>

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 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.
- 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 administration tools 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 nickname schema
- The nickname
- 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.

Target table types

The type of target table depends on how you want your data to appear and on your replication configuration. You can use an existing table as your target, or you can create a new table.

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

Target types

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. SQL 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 SQL 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 topics describe uses for each target type and how you can set the target-table properties to meet your replication needs:

Read-only target tables

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, a history of changes, or a computed summary.

The following topics provide more detail on these types of read-only targets.

User copy and point-in-time targets:

By default, a user copy table will be created as your target type when you define a subscription-set member. Select point-in-time as your target type to keep track of the time at which changes were applied to the target.

User copy

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.

Point in time

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

Base aggregate or change aggregate targets:

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 SQL replication can replicate summary information only rather than each individual row, thus saving both network bandwidth and space in the target table.

Base-aggregate targets

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.

Change-aggregate targets

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.

CCD targets:

You might want to audit the source data or keep a history of how the data is used. By using a consistent-change-data (CCD) table as your target type, you can track the history of source changes.

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 iSeries job name and user profile are available in the UOW table.

Internal CCD targets:

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.

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.

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.

Attributes of internal CCD tables

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 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.

Defining middle tiers in a multi-tier configuration

The basic replication model is a two-tier model, with a single source and one or more targets. You can also set up configurations with three or more tiers.

Restrictions

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.

About this task

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.

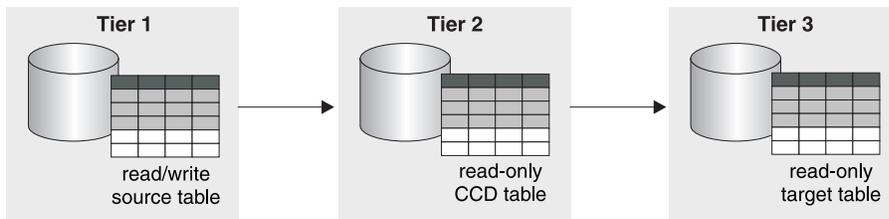


Figure 6. Three-tier replication model. You can replicate data from a source table to a target table, and then from that table to another target table.

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.

Procedure

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. 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). 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 that maps the source table (tier 1) and a CCD target table (tier 2).

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 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). 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). 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)

In update-anywhere replication, changes at the master source table are replicated to dependent target tables, and changes at the replica tables can be replicated back to the master source table.

Before you begin

- 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.
- Non-DB2 databases cannot have replica target-table types and, therefore, cannot participate in update-anywhere replication.

About this task

In update-anywhere replication, the master table and its replicas are read-write tables that all act as both sources and targets.

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. 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.
3. Create a subscription set between the master database and the target database that will contain the one or more replicas.

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.

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, SQL 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.

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.

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 define a subscription-set member to include an existing target table that you defined outside of SQL 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.

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

You can set properties when creating a target table, regardless of type, based on the replication environment that you want.

The following topics explain the common characteristics that you can define for how the source data maps to the target tables.

Replicating a subset of source columns

By default, the target table contains all registered source columns except LOB columns. You might not want to replicate all columns, or the target table might not support all data types defined at the source.

In this case, 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.

Replicating a subset of source rows

By default, the target table contains all the rows in the source table. You might not want to replicate all rows, or you might want to replicate rows containing different sorts of data to different target tables.

You can define a row (horizontal) subset in the subscription-set member 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.

For example, you could define a WHERE clause to replicate all rows for one division of a company. Or you could define a WHERE clause in one subscription-set member to replicate all LOB columns (plus the primary-key column) to one target table, and 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 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.

Examples

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

By default, column names in a target table that is created by SQL replication match the column names in the source table. You can change the names and data lengths of most target columns and still map them to source columns.

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.

Note: Some restrictions exist for mapping LONG VARCHAR columns in DB2 for Linux, UNIX, and Windows to both DB2 for z/OS and DB2 for iSeries.

Using the Replication Center

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.

Mapping to non-DB2 relational tables

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.

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 4. 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 iSeries system commands for replication alter the nickname data type to DATE, so that a DATE is replicated to Oracle and not a TIMESTAMP.

Target key

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.

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.

Options for unique indexes

Your options for creating unique indexes depend on whether you are creating a new target table or using an existing target table.

New target table

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 SQL replication select a unique index for you.

If you do not select columns for the unique index, SQL 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 SQL replication finds one of these definitions for the source table, and the associated columns are registered and part of the target table, SQL replication uses the source table's primary key (or unique index or RRN) as the target key. In the case of a unique constraint, SQL replication creates a unique index for the target table using the constraint columns.

For an iSeries 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.

For target tables on iSeries systems that use the RRN as the target key, you should run the Apply program on an iSeries system to replicate to these target tables.

Existing target table

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.

How the Apply program updates the target key columns with the target-key change option

If you choose the target-key change option when you define a subscription-set member, the Apply program makes special updates to the target key columns when the target key changes.

Prerequisite

In order for the Apply program to update target key columns, the source columns that are part of the target key 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.

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).

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, SQL 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 tries 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
```

Chapter 6. Replicating special data types in SQL replication

When you replicate special data types, such as LOB, 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 SQL replication to work with these data types.

The following topics provide information on replicating special data types:

General data restrictions for replication

SQL replication has specific restrictions for certain data types including data encryption restrictions and data type restrictions.

Data encryption restrictions

SQL replication can replicate some types of encrypted data.

EDITPROC

SQL replication supports DB2 for z/OS source tables that are defined with an edit routine (EDITPROC) to provide additional data security. To use these tables as sources for replication, the DB2 subsystem that contains the tables must be at Version 8 with APAR PK13542.

Scalar functions in DB2 for Linux, UNIX, and Windows

Column data can be encrypted and decrypted using scalar functions in DB2 for Linux, UNIX, and Windows. To use this with replication, the data type must be VARCHAR FOR BIT DATA at the source. This data replicates successfully as long as the source and target use the same code page and the decrypt functions are available. Replication of columns with encrypted data should only be used with servers that support the DECRYPT_BIN or DECRYPT_CHAR function.

Data type restrictions

SQL replication cannot replicate the following data types:

- LOB columns from non-DB2 relational sources
- Any column on which any of the following procedures is defined:
 - FIELDPROC
 - VALIDPROC

SQL 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 for Linux, UNIX, and Windows. See the Alter Table section of the *DB2 SQL Reference* to learn how to enable LONG VARCHAR data.

SQL replication cannot replicate a table that contains abstract data types.

SQL 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) are converted to the base data type in the change-data (CD) table before replication. In addition, if SQL 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.

Large object data types

SQL replication supports large object (LOB) data types, including binary LOB (BLOB), character LOB (CLOB), and double-byte character LOB (DBCLOB).

This topic refers to BLOB, CLOB, and DBCLOB data 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 for Linux, UNIX, and windows, you need DB2 Connect Version 7 or later.
- You cannot refer to LOB data using nicknames.
- Before-image values for LOB 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.

- SQL 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 iSeries.

Chapter 7. Subsetting data in an SQL replication environment

Replication usually involves subsetting. It might involve the choice of certain columns and rows to replicate from a source table when you register a replication source. It might involve the choice of certain registered columns to replicate to each target table when you create subscription sets.

Depending on your replication requirements, you can subset data at the source during registration or at the target during subscription:

- If you have only one target for a source, or if multiple targets need exactly the same data, then it is possible to subset or manipulate data 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.

The following topics provide more detail.

Subsetting data during registration

Certain advanced techniques are useful when subsetting your data before or after it is captured from a registered source. 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 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 fewer rows in the CD table.

This following topics discuss ways to subset data during registration.

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

In some replication scenarios, you might want to prevent certain changes in rows from being captured and replicated to the target tables. To suppress certain changes from being captured, define triggers on your CD tables.

When you register a source, the administration tools let you select which columns you want captured, but they do 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 deletions from the source.

To suppress capture of certain changes, define triggers on your CD tables. These triggers specify what changes the Capture program should ignore, preventing the addition of rows corresponding to changes made in 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). The Capture program ignores any trigger failure that shows an SQLSTATE of 99999 and the row is not inserted into the CD table.

For 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

Subsetting data during subscription can improve replication performance by reducing the amount of data that the Apply program fetches. Fewer rows in the target tables also reduces storage requirements.

The Apply program uses predicates to determine what data to copy during full refresh and change-capture replication. The Replication Center and ASNCLP allow 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 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). You can create the subscription-set member with a PREDICATES value of 'ACCT_BALANCE > 50000'.

You cannot use the Replication Center or ASNCLP 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. Depending on your scenario, you might need to add columns to the update statement to ensure that you update a single row in the IBMSNAP_SUBS_MEMBR table:

```
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 IBMSNAP_SUBS_MEMBR table. Setting this flag ensures that the Apply program joins the UOW and CD tables.

If you want to specify predicates that exceed 1024 bytes (the capacity of the PREDICATES column of the 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 8. Manipulating data in an SQL replication environment

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 any of the following ways:

- Perform data cleansing
- Perform data aggregation
- Populate columns at the target table that do not exist at the source

Use the Apply program to manipulate data, either before or after it applies data to the target, in any of the following ways:

- Using stored procedures or SQL statements
- “Mapping source and target columns that have different names” on page 91
- “Creating computed columns” on page 91

You can manipulate data either before or after it is captured. 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 cycles 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')
```

Enhancing data using stored procedures or SQL statements

When you define subscription set information, you can also 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 data manipulation during replication.

When you define subscription set information, you can also 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 iSeries, 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 processing of each set completes (rather than after processing of a specific set completes).

Mapping source and target columns that have different names

When you are using the Replication Center or ASNCLP command-line program 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. You can also change compatible column attributes.

Also, you can change column attributes (data type, length, scale, precision, and nullability) where they are compatible. You cannot use the replication administration tools to rename columns of existing target tables.

The administration tools try to map columns by name if the target table that is referenced by the subscription-set member exists. If the source and target columns do not match, you can either use the tools 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 or ASNCLP 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 9. Operating the Capture program for SQL replication

This section 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 section.

The following topics provide detail about operating the Capture program.

Starting the Capture program (Linux, UNIX, Windows, and z/OS)

Start the Capture program to begin capturing data from the log for DB2 databases. If you are using trigger-based capture for a non-DB2 relational source, triggers are created at registration and you do not need to start the Capture program.

Before you begin

- Configure connections to the source server and the Capture control server.
- Ensure that you have the proper authorization.
- Create control tables for the appropriate Capture schema.
- Define registrations.
- Configure the Capture and Apply programs.

About this task

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.

When you start the Capture program, you can also specify startup parameters.

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.

Procedure

To start the Capture program on Linux, UNIX, Windows, and z/OS, use one of the following methods:

Method	Description
Replication Center	Use the Start Capture window. To open the window, click the Capture Control Servers folder in the Operations branch of the object tree, and in the contents pane right-click the Capture control server on which the Capture program that you want to start is located. Select Start Capture .
asncap system command	Use this command to start the Capture program and optionally specify startup parameters.

Method	Description
z/OS console or TSO	On z/OS, you can start the Capture program using JCL or as a system-started task. You can specify new invocation parameter values when you start a Capture program with JCL. The best method for specifying invocation parameters when using JCL is to store them in the IBMQREP_CAPPARMS table. The PARM parameter of the EXEC statement cannot have subparameters that exceed 100 characters.
Windows Services	You can create a DB2 replication service on Windows operating systems to start the Capture program automatically when the system is started.

To verify whether a Capture program started, use one of the following methods:

- **z/OS:** If you are running in batch mode, examine the z/OS console or z/OS job log for messages that indicate that the program started.
- Examine the Capture diagnostic log file (*capture_server.capture_schema.CAP.log* on z/OS and *db2instance.capture_server.capture_schema.CAP.log* on Linux, UNIX, and Windows) for a message that indicates that the program is capturing changes.

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".
```

- Check the IBMSNAP_CAPTRACE table for a message that indicates that the program is capturing changes.
- Use the Capture Messages window in the Replication Center to see a message that indicates that the program started. To open the window, right-click the Capture server that contains the Capture program whose messages you want to view and select **Reports** → **Capture Messages**.
- Use the Check Status window in the Replication Center or the asncmd status command to view the status of all Capture threads. To open the window, right-click the Capture server where the Capture program that you want to check is located and select **Check Status**.

Starting the Capture program (iSeries)

Start the Capture program to begin capturing data from the journal.

Before you begin

Before you start the Capture program, 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.

About this task

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.

Procedure

To start the Capture program on iSeries, use one of the following methods:

Method	Description
STRDPRCAP system command (iSeries)	Use the Start DPR Capture (STRDPRCAP) command to start capturing changes.
Replication Center	Use the Start Capture window. To open the window, click the Capture Control Servers folder in the Operations branch of the object tree, and in the contents pane right-click the Capture control server on which the Capture program that you want to start is located. Select Start Capture .

Default operating parameters for the Capture program

When you create the Capture control tables, default values for the Capture program's operating parameters are saved in the IBMSNAP_CAPPARMS table.

The default values are shown in Table 5 and Table 6 on page 96.

Table 5. 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
memory_limit	32 MB	MEMORY_LIMIT
autoprun	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

Table 5. Default settings for Capture operational parameters (Linux, UNIX, Windows, z/OS) (continued)

Operational parameter	Default value	Column name in IBMSNAP_CAPPARMS table
Note:		
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 startup 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.		

Table 6. Default settings for Capture operational parameters (iSeries)

Operational parameter	Default value	Column name in IBMSNAP_CAPPARMS table
CAPCTLLIB	ASN ¹	not applicable
JOBID	*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
RESTART	*YES ³	not applicable

- Note:**
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 stored in the IBMSNAP_CAPPARMS table.
 2. **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.

Descriptions of Capture operating parameters

When you start the Capture program, you can optionally select startup parameters. Here are the startup parameters and recommendations for 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 98
- “capture_path (Linux, UNIX, Windows, z/OS)” on page 98
- “capture_schema (Linux, UNIX, Windows, z/OS)” on page 98
- “capture_server (Linux, UNIX, Windows, z/OS)” on page 99
- “commit_interval (Linux, UNIX, Windows, z/OS)” on page 99
- “lag_limit (Linux, UNIX, Windows, z/OS)” on page 100
- “logreuse (Linux, UNIX, Windows, z/OS)” on page 100
- “logstdout (Linux, UNIX, Windows, z/OS)” on page 100
- “memory_limit (Linux, UNIX, Windows, z/OS)” on page 101
- “monitor_interval (Linux, UNIX, Windows, z/OS)” on page 101
- “prune_interval (Linux, UNIX, Windows, z/OS)” on page 101
- “retention_limit (Linux, UNIX, Windows, z/OS)” on page 102
- “sleep_interval (Linux, UNIX, Windows, z/OS)” on page 103
- “startmode (Linux, UNIX, Windows, z/OS)” on page 103
- “term (Linux, UNIX, Windows, z/OS)” on page 104
- “trace_limit (Linux, UNIX, Windows, z/OS)” on page 104

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 IBMSNAP_CAPTRACE, IBMSNAP_CAPMON, and 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. The Capture program uses the following parameters to determine which rows are old enough to prune:

- **retention_limit** for CD, UOW, and signal tables
- **monitor_limit** for monitor tables
- **trace_limit** for the Capture trace table

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 `//CAPV9`. 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 program. 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.

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 application 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 use a cold start 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 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 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 **autoprun=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.

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 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 IBMSNAP_SIGNAL table.

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 system that is not updated frequently.

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.

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 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=10080

The **trace_limit** specifies how old the rows must be in the IBMSNAP_CAPTRACE table before they are pruned.

When Capture prunes, by default, the rows in the IBMSNAP_CAPTRACE table are eligible to be pruned every 10080 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.

Methods of changing Capture parameters

You can change the saved values of Capture operating parameters, and you can also temporarily override these values when you start the program or while the program is running.

Setting new default values in the IBMSNAP_CAPPARMS table

The 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 IBMSNAP_CAPPARMS table is not set, the default values are used.

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.

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=srddb1 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=srddb1 capture_schema=asnprod chgparms  
commit_interval=15
```

Important: The parameter that you are changing must immediately follow the **chgparms**.

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 (iSeries): 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 iSeries, 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.

Altering the behavior of a running Capture program

You can dynamically change the value of one or more Capture operating parameters. The changes are not saved in the IBMSNAP_CAPPARMS table, but are used until you stop the Capture program or supply new values.

About this task

On Linux, UNIX, Windows, and z/OS, you can change the following Capture parameters while the Capture program is running:

- **autoprun**
- **autostop**
- **commit_interval**
- **lag_limit**
- **logreuse**
- **logstdout**
- **memory_limit**
- **monitor_interval**
- **monitor_limit**
- **prune_interval**
- **retention_limit**
- **sleep_interval**
- **term**
- **trace_limit**

On iSeries, 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.

Procedure

To alter the behavior of a running Capture program, use one of the following methods:

Method	Description
Replication Center	Use the Change Parameters for Running Capture Program window. This method allows you to see the current values of the parameters before changing them. To open the window, open the Operations branch of the object tree, click Capture Control Servers , right click a Capture control server in the contents pane, and click Change Parameters → Running Capture Program .
asncmd chgparms system command (Linux, UNIX, Windows, z/OS)	This method does not show the current values of the parameters.
OVRDPRCAPA system command (iSeries)	Use the Override DPR Capture attributes (OVRDPRCAPA) command to alter the behavior of a running Capture program.

Changing saved operating parameters in the IBMSNAP_CAPPARMS table

The IBMSNAP_CAPPARMS table contains the saved operating parameters for the Capture program. When you start the Capture program, it uses values from this table unless you temporarily override these values using startup parameters or while the program is running.

About this task

Only one row is allowed in the IBMSNAP_CAPPARMS 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 defaults, unless those defaults are overridden by the startup parameters.

The Capture program reads this table only during startup. 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.

Procedure

To change the parameters saved in the IBMSNAP_CAPPARMS table, use one of the following methods:

Method	Description
Replication Center	Use the Change Parameters - Saved window. To open the window, open the Operations branch of the object tree, click Capture Control Servers , right click a Capture control server in the contents pane, and click Change Parameters → Saved .
CHGDPRCAPA system command (iSeries)	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 IBMSNAP_CAPPARMS table.

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.

About this task

iSeries: 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).

Procedure

To stop the Capture program, use one of the following methods:

Method	Description
Replication Center	Use the Stop Capture window. To open the window, open the Operations branch of the object tree, click Capture Control Servers , right click a Capture control server in the contents pane, and click Stop Capture .
asncmd stop system command (Linux, UNIX, Windows, z/OS)	Use this command to stop Capture.
ENDDPRCAP system command (iSeries)	Use the End DPR Capture (ENDDPRCAP) command to stop the Capture program.

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 **autoprun** parameter.

You do not need to stop the Capture program to drop a registration. Always deactivate the registration before you drop it.

Reinitializing Capture

Reinitialize the Capture program if you change any attributes of existing registered objects while the Capture program is running.

About this task

For example, you must reinitialize the Capture program if you change the **CONFLICT_LEVEL**, **CHGONLY**, **RECAPTURE**, **CHG_UPD_TO_DEL_INS** values in the **IBMSNAP_REGISTER** table.

For Capture on iSeries, reinitialize is also needed to start capturing data for a journal that was not being captured previously.

Procedure

To reinitialize the Capture program, use one of the following methods:

Method	Description
Replication Center	Use the Reinitialize Capture window. To open the window, open the Operations branch of the object tree, click Capture Control Servers , right click a Capture control server in the contents pane, and click Reinitialize Capture .
asncmd reinit system command	Use this command to reinitialize Capture.
INZDPRCAP system command (iSeries)	Use the Initialize DPR Capture (INZDPRCAP) command to initialize the Capture program.

Suspending the Capture program (Linux, UNIX, Windows, z/OS)

You can suspend the Capture program to free operating system resources during peak periods without destroying the Capture program environment.

Before you begin

The Capture program with the specific Capture schema must be started.

About this task

You can also 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.

Procedure

To suspend the Capture program, use one of the following methods:

Method	Description
Replication Center	Use the Suspend Capture window. To open the window, open the Operations branch of the object tree, click Capture Control Servers , right click a Capture control server in the contents pane, and click Suspend Capture .
asncmd suspend system command	Use this command to suspend Capture.

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 **autoprun** parameter.

Resuming Capture (Linux, UNIX, Windows, z/OS)

You must resume a suspended Capture program if you want it to start capturing data again.

Procedure

To resume a suspended Capture program, use one of the following methods:

Method	Description
Replication Center	Use the Resume Capture window. To open the window, open the Operations branch of the object tree, click Capture Control Servers , right click a Capture control server in the contents pane, and click Resume Capture .
asncmd resume system command	Use this command to resume Capture.

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.

Chapter 10. Operating the Apply program for SQL replication

The topics in this section describe how to start and stop the Apply program and how to use the ASNDONE and ASNLOAD exit routines.

This section contains the following topics:

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.

Before you begin

Ensure that:

- 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.)

About this task

When you start the Apply program, you can also specify startup parameters.

Procedure

To start the Apply program:

Use one of the following methods:

Option	Description
Replication Center	Use the Start Apply window. To open the window, open the Apply Control Servers folder in the Operations branch of the object tree and click the Apply Qualifiers folder. In the contents pane, right-click the Apply qualifier that represents the Apply program that you want to start and click Start Apply .
asnapply system command (Linux, UNIX, Windows, z/OS)	Use this command to start Apply.
z/OS console or TSO	On z/OS, you can start the Apply program using JCL or as a system-started task. You can specify new invocation parameter values when you start an Apply program with JCL. These values will override the parameter values that are stored in the IBMSNAP_APPPARMS table. The PARM parameter of the EXEC statement cannot have subparameters that exceed 100 characters.
Windows Services	You can create a DB2 replication service on the Windows operating system to start the Q Apply program automatically when the system starts.

After you start the Apply program, it runs continuously (unless you used the **copyonce** startup 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.

To verify whether an Apply program started, use one of the following methods:

- **z/OS:** If you are running in batch mode, examine the z/OS console or z/OS job log for messages that indicate that the program started.
- Examine the Apply diagnostic log file (*apply_server.apply_qualifier.APP.log* on z/OS and *db2instance.apply_server.apply_qualifier.APP.log* on Linux, UNIX, and Windows) for a message that indicates that the program is capturing changes.
- Check the IBMSNAP_APPLYTRACE table for a message that indicates that the program is applying changes.
- Use the Apply Messages window in the Replication Center to see a message that indicates that the program started. To open the window, right-click the Apply qualifier in the contents pane that identifies the Apply program whose messages you want to view and select **Reports** → **Apply Messages**.
- Use the Check Status window in the Replication Center or the asnacmd status command to view the status of all Apply threads. To open the window, right-click the Apply qualifier in the contents pane that identifies the Apply program that you want to check and select **Check Status**.

Starting an Apply program (iSeries)

You can start an instance of the Apply program to begin applying data to your targets.

Before you begin

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

To start an Apply program, use one of the following methods:

Method	Description
STRDPRAPY system command	Use the Start DPR Apply (STRDPRAPY) command to start an Apply program on your local system.
Replication Center	Use the Start Apply window. To open the window, open the Apply Control Servers folder in the Operations branch of the object tree and click the Apply Qualifiers folder. In the contents pane, right-click the Apply qualifier that represents the Apply program that you want to start and click Start Apply .

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.

Default operating parameters for the Apply program

When you create the Apply control tables, default values for the Apply operating parameters are saved in the IBMSNAP_APPPARMS table.

The default values are shown in Table 7 and Table 8 on page 114.

Table 7. 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

Table 7. Default settings for Apply operational parameters (Linux, UNIX, Windows, z/OS) (continued)

Operational parameter	Default value	Column name in IBMSNAP_APPPARMS table
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
spillfile	disk ⁶	SPILLFILE
sleep	y ⁵	SLEEP
sqlerrcontinue	n ³	SQLERRCONTINUE
term	y ⁵	TERM
trlreuse	n ³	TRLREUSE

Note:

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.

Table 8. Default settings for Apply operational parameters (iSeries)

Operational parameter	Description of (*value)
USER (*CURRENT)	The user who signed on to the system.
JOB (*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.

Table 8. Default settings for Apply operational parameters (iSeries) (continued)

Operational parameter	Description of (*value)
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 IBMSNAP_APPLYTRAIL table when the Apply program starts.
OPTSNGSET (*NO)	Do not optimize performance of the Apply program for processing a single subscription set.

Descriptions of Apply operating parameters

When you start the Apply program, you can optionally select startup parameters. Here are the startup parameters and recommendations for 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 116
- “control_server (Linux, UNIX, Windows, z/OS)” on page 116
- “copyonce (Linux, UNIX, Windows, z/OS)” on page 116
- “db2_subsystem (z/OS)” on page 117
- “delay (Linux, UNIX, Windows, z/OS)” on page 117
- “errwait (Linux, UNIX, Windows, z/OS)” on page 117
- “inamsg (Linux, UNIX, Windows, z/OS)” on page 118
- “loadxit (Linux, UNIX, Windows, z/OS)” on page 118
- “logreuse (Linux, UNIX, Windows, z/OS)” on page 118
- “logstdout (Linux, UNIX, Windows, z/OS)” on page 119
- “notify (Linux, UNIX, Windows, z/OS)” on page 119
- “opt4one (Linux, UNIX, Windows, z/OS)” on page 119
- “pwdfile (Linux, UNIX, Windows, z/OS)” on page 119
- “sleep (Linux, UNIX, Windows, z/OS)” on page 120
- “spillfile (Linux, UNIX, Windows, z/OS)” on page 120
- “sqlerrcontinue (Linux, UNIX, Windows, z/OS)” on page 121
- “term (Linux, UNIX, Windows, z/OS)” on page 121
- “trlreuse (Linux, UNIX, Windows, z/OS)” on page 122

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.

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 `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.

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.

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, z/OS)

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.

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 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 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.

Methods of changing Apply operating parameters

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 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 for the Apply program.

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'
```

Changing saved Apply parameters in the IBMSNAP_APPPARMS table (Linux, UNIX, Windows, z/OS)

The IBMSNAP_APPPARMS table contains the saved operating parameters for the Apply program. When you start the Apply program, it uses values from this table unless you temporarily override these values using startup parameters.

About this task

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 startup 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.

Stopping the Apply program

When you stop the Apply program, it no longer copies data to the target tables, and it updates the control tables to ensure that the program starts cleanly the next time that you start it.

Procedure

To stop the Apply program:

Use one of the following methods:

Option	Description
Replication Center	Use the Stop Apply window. To open the window, open the Apply Control Servers folder in the Operations branch of the object tree and click the Apply Qualifiers folder. In the contents pane, right-click the Apply qualifier that represents the Apply program that you want to start and click Stop Apply .
asnacmd stop system command (Linux, UNIX, Windows, z/OS)	Use this command to stop Apply.
ENDDPRAPY system command (iSeries)	Use the End DPR Apply (ENDDPRAPY) command to stop an Apply program on your local system.

Modifying the ASNDONE exit routine (Linux, UNIX, Windows, z/OS)

You can customize the ASNDONE exit routine on Linux, UNIX, Windows, and z/OS operating systems to modify the behavior of the Apply program after it finishes processing subscriptions.

About this task

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 IBMSNAP_APPLYTRAIL table. To deactivate the subscription set, configure the exit routine so that it sets ACTIVATE=0 in the 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 (iSeries)

You can customize the ASNDONE exit routine on iSeries operating systems to modify the behavior of the Apply program after it finishes processing subscriptions.

About this task

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

IBMSNAP_APPLYTRAIL table. To deactivate the subscription set, configure the exit routine so that it sets ACTIVATE=0 in the IBMSNAP_SUBS_SET table.

- Use the exit routine to manipulate data after it is applied for each subscription set. (You can also 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 exit routine to meet your requirements. Table 9 indicates where you can find the source code for this routine in C, COBOL, and RPG languages:

Table 9. Source code for ASNDONE

Compiler language	Library name	Source file name	Member name
C	QDP4	QCSRC	ASNDONE
COBOL	QDP4	QCBLESRC	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 iSeries 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

You can use the ASNLOAD exit routine to perform a full refresh of target tables more efficiently than the Apply program's normal method of loading data into targets.

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.

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 iSeries 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.

The following topics describe how to use the ASNLOAD exit routine on various operating systems.

Refreshing target tables with the ASNLOAD exit routine (Linux, UNIX, Windows)

You can use the ASNLOAD exit routine to refresh target tables more efficiently on Linux, UNIX, and Windows operating systems. You can also modify the routine before using it.

Before you begin

- The target-table columns must match both the order and data type of the source tables.
- The target table must contain only columns that are part of the replication mapping.
- The user ID that runs Apply must be the user ID for the DB2 instance where ASNLOAD runs. For example, on Linux and UNIX, make sure that both the DB2 instance and Apply user ID are members of a common group. Next, set the permission bits for the Apply starting directory to provide write access for the DB2 instance using the `chmod 775` command.

Restrictions

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 a nickname, or if the target database is the same as the source database. LOAD FROM CURSOR 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 column must be set to indicate the LOAD FROM CURSOR function will be used.
 - The LOADX_SRC_N_OWNER and LOADX_SRC_N_TABLE columns must specify the source nickname information for the subscription-set member that includes the source table.

About this task

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. The routine can use the DB2 EXPORT utility with either the DB2 IMPORT utility or the DB2 LOAD utility, or it can use the LOAD FROM CURSOR utility.

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 IBMSNAP_SUBS_MEMBR table or by updating a sample configuration file (asnload.ini).

To use the ASNLOAD routine as provided, start the Apply program using the **loadxit=y** parameter.

Procedure

To use a modified version of the ASNLOAD 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, the exit will attempt to connect without the **user** or **using** parameters.

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.
4. Start the Apply program with the loadxit=y parameter to call the ASNLOAD exit routine.

The ASNLOAD exit routine generates the following files 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)

You can use the ASNLOAD exit routine to refresh target tables more efficiently on z/OS operating systems. You can also modify the routine before using it.

Before you begin

- The target-table columns must match both the order and data type of the source tables.
- The target table must contain only columns that are part of the replication mapping.

About this task

The ASNLOAD exit routine calls the LOAD FROM CURSOR utility that is available with the DB2 V7 (or higher) Utilities Suite. The utility 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.

To use the ASNLOAD routine as provided, start the Apply program using the **loadxit=y** parameter.

Procedure

To use a modified version of the ASNLOAD exit routine:

1. Modify the 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.
 - 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 UDB for z/OS V8 Utility Guide and Reference*.

- 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. The routine 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.

The ASNLOAD exit routine generates the following files 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)

In addition to customizing the exit code itself, you can customize the behavior of the ASNLOAD exit routine by either updating columns in the IBMSNAP_SUBS_MEMBR table or by updating a configuration file.

The following topics describe these options:

Using the IBMSNAP_SUBS_MEMBR table to set ASNLOAD options

You can use columns in the IBMSNAP_SUBS_MEMBR table to customize the behavior of the ASNLOAD exit routine.

About this task

Use the LOADX_TYPE column to specify a load option. The valid values for LOADX_TYPE are:

null (default)

z/OS: Use the LOAD FROM CURSOR utility.

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` utility.

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 `IBMSNAP_SUBS_MEMBR` table.

If you are replicating from a DB2 table to another DB2 table and the source and 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` utility.

- 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.

About this task

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 directory specified by the `apply_path` parameter.

Procedure

To use the ASNLOAD configuration file:

1. Edit the sample file `sqllib/samples/repl/asnload.ini`.
2. Store the file in the directory specified by the `apply_path` parameter for the Apply instance that invoked the ASNLOAD exit routine.

Refreshing target tables with the ASNLOAD exit routine (iSeries)

You can use the ASNLOAD exit routine to refresh target tables more efficiently on iSeries. You can also modify the routine before using it.

Before you begin

- The target-table columns must match both the order and data type of the source tables.
- The target table can only contain columns that are part of the replication mapping.

About this task

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.

To use the ASNLOAD exit routine as provided, start the Apply program using the FULLREFPGM parameter.

Procedure

To use a modified version of the ASNLOAD exit 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, as shown in Table 10.

Table 10. Source code for ASNLOAD

Compiler language	Library name	Source file name	Member name
C	QDP4	QCSRC	ASNLOAD
COBOL	QDP4	QCBLESRC	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.

Chapter 11. Operating the replication programs (z/OS)

The following topics describe operating the replication programs on the z/OS operating system.

Using system-started tasks to operate the replication programs

You can use system-started tasks to operate the Capture program, Apply program, and the Replication Alert Monitor.

Procedure

To use system started tasks to operate the replication programs, use this example from the Capture program:

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,
//PARAM='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=
//SYSTEM DD DUMMY
//
```

Using JCL to operate replication programs

On z/OS, you can use JCL to start, stop, and modify running replication programs. This allows you to save scripts if you will perform the operation repeatedly.

About this task

The SQL replication V9 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.

Procedure

To operate the replication programs with JCL:

1. Start the replication programs.

Option	Description
Start the Capture program with a batch job.	Prepare the JCL for z/OS by specifying the appropriate optional invocation parameters in the PARM field of the ASNSTRC batch job. Run the job from TSO or the z/OS console. You can find the job in the SASNSAMP sample library.
Start the Apply program with a batch job.	Prepare the JCL for z/OS by specifying the appropriate optional invocation parameters in the PARM field of the ASNSTRA batch job. Run the job from TSO or the z/OS console. You can find the job in the SASNSAMP sample library.
Start the Replication Alert Monitor with a batch job.	Prepare the JCL for z/OS by specifying the appropriate optional invocation parameters in the PARM field of the ASNSTRM batch job. Run the job from TSO or the z/OS console. You can find the job in the SASNSAMP sample library.
Start the Replication Alert Monitor with JCL.	<p>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.</p> <p>An example of this line in the invocation JCL is:</p> <pre>//monasn EXEC PGM=ASNMON,PARM='monitor_server=DSN monitor_qual=monqual '</pre> <p>where DSN is a subsystem name and monqual is the monitor qualifier.</p>

2. **Optional:** Modify replication programs that have already started.

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:

►► F—*jobname*—, —| Parameters |—————►►

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*.

Starting the Apply program on z/OS with JCL

You can start the Apply program on z/OS by modifying and running a prepared sample script from your samples directory.

Procedure

To start the Apply program on z/OS with JCL:

1. Prepare the JCL for z/OS by specifying the appropriate invocation parameters in the PARM field of the Apply job.
2. 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'
```

3. Submit the JCL from TSO or from the MVS console.

Starting the Capture program on z/OS with JCL

You can start the Capture program on z/OS by modifying and running a prepared sample script from your samples directory.

Procedure

To start the Capture program on z/OS with JCL:

1. Prepare the JCL for z/OS.
 - a. Specify the appropriate optional invocation parameters in the PARM field of the Capture job.
 - b. 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 *WebSphere Information Integration Replication Installation and Customization Guide for z/OS*.

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.

Using MVS Automatic Restart Manager (ARM) to automatically restart replication

The Capture program, the Apply program, and the Replication Alert Monitor can be used with the MVS Automatic Restart Manager (ARM).

Before you begin

Ensure that ARM is installed and that the replication programs are set up correctly. 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.)

About this task

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*.

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.

Before you begin

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.

About this task

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 IBMSNAP_RESTART table. It converts the non-data sharing log sequence numbers (RBA) to the equivalent sequence numbers (LRSN) in a data-sharing environment.

Procedure

To run the ASNPLXFY utility in the USS data-sharing environment:

1. Stop the Capture program.
2. Issue the ASNPLXFY command from a command line. Here is an example:

```
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 12. Changing an SQL replication environment

The following topics explain procedures and issues and for making day-to-day changes to a Q replication environment.

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.

About this task

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.

Procedure

To register new objects:

Use one of the following methods to register new objects:

Method	Description
ASNCLP command-line program	Use the CREATE REGISTRATION command to register a source table, view, or nickname. For example, the following commands set the environment and register the DEPARTMENT table in the DB2 SAMPLE database for full refresh replication. <pre>SET SERVER CAPTURE TO DB SAMPLE; SET OUTPUT CAPTURE SCRIPT "registernew.sql"; SET LOG "registernew.err"; SET RUN SCRIPT LATER; CREATE REGISTRATION (DB2ADMIN.DEPARTMENT) FULL REFRESH ONLY;</pre>
Replication Center	Use one of the following windows: <ul style="list-style-type: none">Registered Table Properties notebookRegistered View Properties notebookRegistered Nickname Properties notebook To open the windows, click the Registered Tables , Registered Views , or Registered Nicknames folder in the object tree under a Capture control server, right-click the registered object in the contents pane, and select Properties .
ADDDPRREG system command (iSeries)	Use the Add DPR registration (ADDDPRREG) command to register a new table on iSeries.

Changing registration attributes for registered objects

You can change the registration attributes of existing registered objects at any time.

Procedure

To change registration attributes for registered objects:

1. Change the attributes by using one of the following methods.

Method	Description
ASNCLP command-line program	Use the ALTER REGISTRATION command to change the properties of a registered object. For example, the following commands set the environment and change the registration for the STAFF table in the DB2 SAMPLE database so that updates are captured as delete-insert pairs: <pre>SET SERVER CAPTURE TO DB SAMPLE; SET OUTPUT CAPTURE SCRIPT "register.sql"; SET LOG "register.err"; SET RUN SCRIPT LATER; ALTER REGISTRATION (DB2ADMIN.STAFF) UPDATE AS DELETE INSERT ON;</pre>
Replication Center	Use one of the following windows: <ul style="list-style-type: none">• Registered Table Properties notebook• Registered View Properties notebook• Registered Nickname Properties notebook To open the windows, click the Registered Tables , Registered Views , or Registered Nicknames folder in the object tree under a Capture control server, right-click the registered object in the contents pane, and select Properties .

2. After you change the attributes, reinitialize the Capture program.

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.

Before you begin

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.

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 recreate the existing registered source.

About this task

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.

Procedure

To add columns to source tables:

1. Quiesce all activity against the source table that you want to alter.
2. Stop the Capture program.
3. **Optional:** If you need to keep the Capture program active during this procedure, insert a USER signal in the 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.
4. Deactivate all subscription sets that subscribe to this source table from the Replication Center.

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.

5. Submit an ALTER TABLE ADD statement in SQL to add the new columns to the source table.
6. Add the new columns to the CD table by using the ALTER REGISTRATION command in the ASNCLP command-line program or the Registered Table Properties notebook in the Replication Center. 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.
7. Submit an ALTER TABLE ADD statement in SQL to add the new columns to the target table.
8. Deactivate any associated subscription sets that you did not already deactivate from the Replication Center. 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.

9. Add the new columns to the associated subscription-set members by using the ALTER MEMBER ADD COLS command in the ASNCLP command-line program or the Add Column to Target Table window in the Replication Center.
10. On UNIX, Windows, z/OS, if you are running the Apply program with **opt4one** set to *y*, stop and then restart the Apply program.
11. Reactivate the subscription sets.

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.

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.

About this task

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 SQL 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.

Procedure

To deactivate a registered object:

1. Deactivate all associated subscription sets using the Replication Center. Click the **Subscription Sets** folder, right-click the active subscription sets in the contents pane and select **Deactivate**.

2. Deactivate the registered object using one of the following methods:

Method	Description
Replication Center	Click the Registered Tables folder, right-click the registered table in the contents pane and select Stop Capturing Changes .
SQL	Manually insert a CAPSTOP signal into the IBMSNAP_SIGNAL table.

Making registrations eligible for reactivation

When you reactivate a registration, 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.

Before you begin

Read the error messages that were generated by the Capture program regarding any deactivated registrations.

Familiarize yourself with the structure of the Capture control tables and with the Capture programs running on your system.

About this task

Unexpected errors can cause the Capture program to set the value of the STATE column to S (Stopped) in the 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.

Procedure

To correct unexpected errors and make registration eligible for reactivation:

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.

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 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, SQL 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.

Before you begin

- Deactivate the registration to ensure that the Capture program finishes any current processing of this object.
- Deactivate the subscription sets associated with the source.

About this task

Important: Deactivation is an asynchronous process. Be sure that the deactivation process finishes before you remove the object.

If changes are made while the Capture program is running, these changes will not be recognized by the Capture program until you either reinitialize, or stop and restart the Capture program.

Procedure

To remove registrations, use one of the following methods:

Method	Description
ASNCLP command-line program	Use the DROP REGISTRATION command to drop one or more registrations. For example, the following commands set the environment and drop the registration for the DEPARTMENT table in the DB2 SAMPLE database: <pre>SET SERVER CAPTURE TO DB SAMPLE; SET OUTPUT CAPTURE SCRIPT "dropregis.sql"; SET LOG "dropregis.err"; SET RUN SCRIPT LATER; DROP REGISTRATION (DB2ADMIN.DEPARTMENT);</pre>

Method	Description
Replication Center	Use the Delete Registered Tables or Delete Registered Views windows. To open the windows, click the Registered Tables or Registered Views folder in the object tree under a Capture control server, right-click the registered object in the contents pane, and select Delete .
RMVDPRREG system command (iSeries)	Use the Remove DPR registration (RMVDPRREG) command to remove a single source table from the IBMSNAP_REGISTER table.

Changing Capture schemas

You can change an existing Capture schema.

Before you begin

- Familiarize yourself with the SQL replication control tables and with the subscription sets that are defined on your system.
- Determine the new Capture schema name.
- 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 or later.

Restrictions

You should not use this procedure if your source server is a non-DB2 relational database.

About this task

Tip: 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, recreate the monitoring definitions with the new Capture schema name. Then, you can reinitialize the associated monitors using the `asnmcmd reinit system` command. You can also stop the monitors by using the `asnmcmd stop system` command and then restart the programs using the `asnmon system` command.

Procedure

To change capture schemas:

1. Create control tables for a new Capture schema.
2. Stop the Capture program.
3. Deactivate all associated subscription sets by using the Replication Center.
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 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 (iSeries 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. Drop the existing schema and its associated Capture control tables using the Replication Center or ASNCLP.
8. Restart the Capture program with the new schema name.
9. Reactivate the associated subscription sets using the Replication Center.

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.

Before you begin

Before you create a new subscription set, register the tables or views that you want to use as sources.

Restrictions

If the corresponding Apply program is active, do not activate the new subscription set until the subscription set is fully defined.

About this task

This procedure addresses the addition of a new subscription set, with or without subscription-set members.

Procedure

To create a new subscription set, use one of the following methods:

Method	Description
ASNCLP command-line program	Use the CREATE SUBSCRIPTION SET command to create an empty set.
Replication Center	Use the Create Subscription Set notebook to create a set and add a member or to create an empty set. To open the notebook, expand the Apply control server where the set will be defined, right click the Subscription Sets folder and click Create .
ADDDPRSUB system command (iSeries)	Use the Add DPR subscription set (ADDDPRSUB) command to create a subscription set with either one member or no members.

Adding new subscription-set members to existing subscription sets

You can add one or more members that each use the same source table to one or more existing subscription sets. For example, if you select three subscription sets, you can add one member to each of those subscription sets, all of them using the same replication source.

About this task

When you add a member to a subscription set, you are inserting information about the new member into the Apply control tables. In most cases, the Apply program will read this information at the beginning of the next Apply cycle.

However, if you add a member to a subscription set that is being processed with the OPT4ONE option on Linux, UNIX, Windows, or z/OS or with the OPTSNGSET option on iSeries, you must stop the Apply program for the subscription set and then restart it. If you process a set with the OPT4ONE option, the Apply program reads into memory the control table information for the set so that it does not need to go to the control tables to read the information for the set at the beginning of each Apply cycle.

If the source table for the member is registered for differential replication and the Capture program is already running, you do not need to stop or reinitialize the Capture program before you add the member. Because the added member must use a registered table as its source, the Capture program will already be capturing changes for it.

Procedure

To add new subscription-set members to existing subscription sets, use one of the following methods:

Method	Description
ASNCLP command-line program	Use the CREATE MEMBER command to add a subscription-set member to an existing subscription set.
Replication Center	Use the Add Member to Subscription Set notebook. To open the notebook, click the Registered Tables folder. In the contents pane, right-click a registered table that you want to use and click Add Member .
ADDDPRSUBM system command (iSeries)	Use the Add DPR subscription-set member (ADDDPRSUBM) command to add a member to an existing subscription set.

Disabling subscription-set members from existing subscription sets

If you want the Apply program to ignore a failing subscription-set member and continue processing the rest of the subscription set, you must disable the failing subscription-set member.

About this task

If there is a problem replicating to a table in the subscription set, the Apply program inserts an error messages into the IBMSNAP_APPLYTRAIL table and continues processing other members in the Apply cycle.

Procedure

To disable a subscription-set member, issue the following SQL UPDATE statement:

```
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).

Procedure

To re-enable a subscription-set member, issue the following SQL UPDATE statement:

```
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 properties of subscription sets

You can change the properties of a subscription set while Apply continues to run and process other sets, and then reactivate the set before the next Apply cycle.

About this task

The following list describes attributes that you might need to change:

- 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)

By first deactivating the subscription set, you prevent the Apply program from processing the set while you enter your changes. The Apply program recognizes your subscription set changes during the next Apply cycle after you reactivate the set.

Procedure

To change the properties of a subscription set:

1. Deactivate the subscription set by using the Replication Center.
2. Use one of the following methods to change the subscription set:

Method	Description
ASNCPL command-line program	<p>Use the ALTER SUBSCRIPTION SET command.</p> <p>The following commands set the environment and change the subscription set SET00 to lower the timing interval to 15 minutes:</p> <pre> SET SERVER CAPTURE TO DB SAMPLE; SET SERVER CONTROL TO DB TARGET; SET OUTPUT CAPTURE SCRIPT "capsubsetchg.sql" CONTROLSRIPT "appsubsetchg.sql"; SET LOG "subsetchg.err"; SET RUN SCRIPT LATER; ALTER SUBSCRIPTION SET SETNAME SET00 APPLYQUAL AQ00 SETTYPE R ACTIVATE YES TIMING INTERVAL 15 COMMIT COUNT NULL; </pre>
Replication Center	<p>Use the Subscription Set Properties notebook. To open the notebook, click the Subscription Sets folder within an Apply control server, right-click the subscription set in the contents pane and click Properties.</p>

3. Reactivate the subscription set.

If you set the **opt4one** Apply program parameter to *y*, stop and then restart the Apply program or your changes will not be recognized (UNIX, Windows, z/OS).

Changing subscription set names

You can change the name of a subscription set without having to drop and re-create the subscription set and all of its members.

Before you begin

Before running these SQL statements, familiarize yourself with the structure of the SQL replication control tables and with the subscription sets defined on your system.

Tip: If you set up monitoring definitions or started Replication Alert Monitor programs to detect alert conditions for the subscription set, drop these definitions. After you change the subscription-set name, re-create the monitoring definitions through the Replication Center or ASNCLP. Then, you can reinitialize the monitors using the `asnmcmd reinit system` command. You can also stop the monitors using the `asnmcmd stop` command and then restart the programs using the `asnmcmd` command.

Procedure

To change the name of a subscription set:

1. Use the Replication Center to deactivate the subscription set.
2. From the Apply control server, run the following SQL statements to change the name of the subscription set in the `IBMSNAP_SUBS_SET`, `IBMSNAP_SUBS_MEMBR`, and `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 `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 IBMSNAP_PRUNE_SET and 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. If you are running the Apply program on Linux, UNIX, Windows, or z/OS, with **opt4one** set to *y*, stop and then restart the Apply program.
6. Reactivate the subscription set from the Replication Center.

Splitting a subscription set

You can split a subscription set into two or more sets without having to remove and re-create subscription set information.

Before you begin

- Before running these SQL statements, familiarize yourself with the structure of the SQL 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.

About this task

Tip: If you set up monitoring definitions or started Replication Alert Monitor programs to detect alert conditions for the subscription set, drop these definitions. After you split the subscription-set, re-create the monitoring definitions through the Replication Center or ASNCLP. Then, you can reinitialize the monitors using the `asnmcmd reinit system` command. You can also stop the monitors using the `asnmcmd stop` command and then restart the programs using the `asnmon` command.

Procedure

To split a subscription set:

1. Deactivate the subscription set that you want to split from the Replication Center. From the Subscription Sets folder, right-click the active subscription set in the contents pane and select Deactivate.
2. Create a new subscription set. The new set is represented by a new row in the 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 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 IBMSNAP_SUBS_MEMBR table and the 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 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,..Stmnt);

```

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 *Stmntn* 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 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. If you are running the Apply program with **opt4one** set to y on Linux, UNIX, Windows or z/OS, stop and then restart the Apply program.
10. Reactivate both subscription sets from the Replication Center.

Merging subscription sets

You can merge two subscriptions sets into one. 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 recreate subscription set information.

Before you begin

Before running these SQL statements, familiarize yourself with the structure of the SQL 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.

Restrictions

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.

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.

Procedure

To merge subscription sets:

1. Stop the associated Capture program. Wait until both subscription sets reach the same synchpoint and synctime as indicated in the IBMSNAP_SUBS_SET table.

Tip: If you do not want to stop the Capture program, insert a USER signal in the IBMSNAP_SIGNAL table, and generate an event with the END_SYNCHPOINT (in the 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. Deactivate both subscription sets from the Replication Center.
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 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 IBMSNAP_SUBS_MEMBR and 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 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 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. If you are running the Apply program on Linux, UNIX, Windows, or z/OS with **opt4one** set to y stop and then restart the Apply program.
9. Reactivate the merged subscription set from the Replication Center.

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.

Before you begin

Before running these SQL statements, familiarize yourself with the structure of the SQL 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.
- 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.

About this task

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 workloads of the Apply programs.

Tip: If you set up monitoring definitions or started Replication Alert Monitor programs to detect alert conditions for the Apply qualifier, drop these definitions. After you change the qualifier, re-create the monitoring definitions through the Replication Center or ASNCLP. Then, you can reinitialize the monitors using the `asnmcmd reinit system` command. You can also stop the monitors using the `asnmcmd stop` command and then restart the programs using the `asnmon` command.

You must run the SQL statements in this procedure for each subscription set that you want to move.

Procedure

To change Apply qualifiers of subscription sets:

1. Deactivate the subscription sets that you want to change using the Replication Center.
2. From the Apply control server, run the following SQL statements to change the Apply qualifier of the subscription set in the `IBMSNAP_SUBS_SET`, `IBMSNAP_SUBS_MEMBR`, and `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 statements at the Apply control server to change the Apply qualifier of the subscription set in the 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 IBMSNAP_PRUNE_SET and 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. If you are running the Apply program with **opt4one** set to y on Linux, UNIX, Windows or z/OS, stop and then restart the Apply program.
7. Reactivate the subscription sets using the Replication Center.

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.

Before you begin

Before running these SQL statements, familiarize yourself with the structure of the SQL replication control tables and with the subscription sets defined on your system.

About this task

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.

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.

Procedure

1. From the Replication Center, deactivate the set. Click the **Subscription Sets** folder, right-click the active subscription set in the contents pane and select **Deactivate**.
2. From the Capture control server, run the following SQL statements to reset the pruning information in the IBMSNAP_PRUNE_SET and 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.

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

To remove subscriptions sets:

1. To ensure that the Apply program has completed any current processing for the subscription set, deactivate the subscription set before you remove it, from the Replication Center Click the **Subscription Sets** folder, right-click the active subscription set in the contents pane and select **Deactivate**.
2. Use one of the following methods to remove a deactivated subscription set:

Method	Description
ASNCLP command-line program	<p>Use the DROP SUBSCRIPTION SET command.</p> <p>The following commands set the environment and drop a subscription set named SET00 with an Apply qualifier of AQ00.</p> <pre> SET SERVER CAPTURE TO DB SAMPLE; SET SERVER CONTROL TO DB TARGET; SET OUTPUT CAPTURE SCRIPT "drpcapsubset.sql" CONTROLSRIPT "drpappsubset.sql"; SET LOG "drpsubset.err"; SET RUN SCRIPT LATER; DROP SUBSCRIPTION SET SETNAME SET00 APPLYQUAL AQ00; </pre>
Replication Center	Use the Delete Subscription Set window. To open the window, click the Subscription Sets folder, right-click the active subscription set in the contents pane and select Delete .
RMVDPRSUB system command (iSeries)	Use the Remove DPR subscription set (RMVDPRSUB) command to remove a subscription set.

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 `IBMSNAP_SIGNAL` table. These 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.

About this task

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 to insert an `END_SYNCHPOINT` value in the `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.

When to use 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.

You can use the Capture CMD STOP signal in the following situations:

- 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 source table changes:

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');
```

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 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.

Before you begin

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_SYNCPOINT 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).

iSeries: 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 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.

About this task

The Apply program initiates the handshake by inserting a CMD type CAPSTART subtype signal into the 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 outside the Apply program:

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 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.
 3. **Optional for iSeries:** 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.

Adjusting for Daylight Savings Time (iSeries)

On iSeries, the Capture program uses a timestamp and the journal sequence number when reading changes from a journal. This process can create problems when it is necessary to adjust the system clock for U.S. Daylight Savings Time in the autumn and spring.

About this task

iSeries systems provide two methods for adjusting to Daylight Savings Time:

V5R3 The system either slows down its clock (autumn) or speeds up (spring) to avoid skipping or duplicating any timestamps. If you are running the Capture program on iSeries V5R3 and use this new method to make the time change, you do not need to use the procedure below.

Before V5R3

You must stop all activity on the system for one hour and then move the clock back one hour in autumn. With this method, you need to use the procedure below.

Procedure

To do adjust for Daylight Savings Time:

1. Follow these steps when you must turn back the clock by one hour in autumn:
 - a. Stop the Capture program and any applications that update the source tables.
 - b. Wait for the system time to move forward by at least an hour without any new journal entries to the source journal.
 - c. Set the system time back by an hour.
 - d. Restart the Capture program.

The Capture program restarts with a timestamp that is lower than the current system time. No journal entries will be added until the new system time is greater than the system time just before the time change, so there is no possibility of missing any data.

Recommendation: Although the time change has no effect on the Apply program, stop and restart the Apply program during the time change also.

2. Follow this procedure when you must move the clock forward by one hour in spring:
 - a. Stop the Capture program and make the time change. The Capture program responds as though an hour passed with no changes to the source tables.

Options for 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.

The promote 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.

The following list describes the three options for promoting your replication configuration to another system.

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 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 iSeries, Linux, UNIX, Windows, and z/OS operating systems. The promote functions copy replication definitions between like systems only, for example from one DB2 for z/OS system to another DB2 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 iSeries remote journals.

Chapter 13. Maintaining a SQL replication environment

You should maintain the source systems, control tables, and target tables that reside on your database and are used by SQL replication.

SQL 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.

The following topics discuss the maintenance requirements of source systems, control tables, and target tables.

Maintaining 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 iSeries systems), the log data used by the Capture program, and any Capture triggers that are used on non-DB2 relational database sources.

These topics explain how to maintain your source tables and log files properly and how to ensure that these tables and files are always accessible to SQL replication.

Access to source tables and views

You need to consider the availability of source tables to SQL replication so that the Capture and Apply programs are always able to proceed.

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.

SQL replication does not require direct access to source tables during most replication processing. However, SQL 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 the Apply program 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.

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 SQL 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: SQL replication does not use log data from non-DB2 relational databases.

Retaining log data (Linux, UNIX, Windows)

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.

Before you begin

Note: You must configure your database to use user-exit archiving for your Capture programs to retrieve data from archived logs.

About this task

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 SQL 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:

1. Run the following SQL statement to obtain the MIN_INFLIGHTSEQ value from the IBMSNAP_RESTART table:

For partitioned databases: 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.

```
SELECT MIN_INFLIGHTSEQ
FROM ASN.IBMSNAP_RESTART
WITH UR;
```

The MIN_INFLIGHTSEQ value appears. 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
```

Attention: The Capture program updates the IBMSNAP_RESTART each time it commits data, based on the value of the **commit_interval** parameter. Because the SELECT statement that is used in this procedure specifies an uncommitted read (UR), you might receive an uncommitted value for MIN_INFLIGHTSEQ. To ensure that you have the most accurate value, run the SELECT statement, wait for the commit interval to elapse, and then run the SELECT again. Use the lower value for MIN_INFLIGHTSEQ for the rest of this procedure.

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
```

Access to journal receivers (iSeries)

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.

Considerations for managing 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 that is stored in a compressed table space. DB2 uses the current compression dictionary for decompression. In some cases the compression dictionary may be unavailable. The Capture program takes different actions in each case:

If the compression dictionary is temporarily unavailable

DB2 returns an error to the Capture program. The Capture program makes several attempts to continue processing. If the dictionary remains unavailable, the Capture program issues an ASN0011E message and terminates.

If the compression dictionary is permanently unavailable

A compression dictionary may be lost if you use the REORG utility without specifying KEEPDICTIONARY=YES. In this case, the Capture program follows the error action that is specified by the STOP_ON_ERROR option for the registration. If STOP_ON_ERROR=N (no), Capture deactivates the registration. If STOP_ON_ERROR=Y (yes), the Capture program issues an ASN0011E message and terminates.

With APAR PK19539 (DB2 for z/OS Version 8), DB2 will keep one backup of the compression dictionary in memory when you use the REORG utility without specifying KEEPDICTIONARY=YES. So you do not need to specify KEEPDICTIONARY=YES unless:

- You restart DB2.
- You use the REORG utility twice for the same tablespace before the Capture program reads all of the old log records for that table.

To avoid these situations in DB2 for z/OS Version 7, let the Capture program process all log records for a table before performing any activity that affects the compression dictionary for that table. Some of the following activities can affect compression dictionaries:

- 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

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 control tables

SQL 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 tables changes frequently during normal processing:

- IBMSNAP_APPLY_JOB (iSeries)
- IBMSNAP_APPLYTRACE
- IBMSNAP_APPLYTRAIL
- IBMSNAP_CAPMON
- IBMSNAP_CAPTRACE
- CD tables
- CCD tables
- IBMSNAP_ALERTS
- IBMSNAP_MONTRACE
- IBMSNAP_MONTRAIL
- IBMSNAP_SIGNAL
- BMSNAP_SUBS_EVENT
- 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.

The RUNSTATS utility for SQL replication (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 SQL replication. However, you should run the RUNSTATS utility on your change-data (CD), 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)

Binding your packages and plans with the isolation level set to UR (uncommitted reads) ensures optimal system performance.

Many of the SQL replication packages and plans are bound using isolation UR. 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. SQL replication packages must remain bound with isolation UR to maintain optimal system performance.

Reorganizing your control tables

You should regularly reorganize dynamic control tables that are frequently updated.

About this task

Your CD and IBMSNAP_UOW tables receive many INSERTS during change capture and many DELETES during pruning. The size of the IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_APPLYTRAIL tables can change dramatically depending on the update rates of your replication source tables.

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 other control tables.

Procedure

To reorganize your control tables, use one of the following methods:

Method	Description
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 (iSeries)	You can reorganize the UOW table and active CD tables when the Capture program ends by specifying the RGZCTLTLBL(*YES) parameter on the ENDDPRCAP command.
REORG command (Linux, UNIX, Windows)	Use this command to eliminate fragmented data and reclaim space.

Pruning dynamic control tables maintained by the Capture programs (Linux, UNIX, Windows, z/OS)

You can manually or automatically prune tables that fluctuate in size.

About this task

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 (iSeries 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.

Procedure

To prune dynamic control tables that are maintained by the Capture program:

1. If you want to prune the dynamic control tables automatically, set the **autoprun** parameter to yes using one of the following methods:

Method	Description
Start a Capture program with automatic pruning.	Issue the asncap system command with autoprun=y . Set the prune_interval parameter to specify how frequently the automatic pruning process occurs.
Enable automatic pruning for a running Capture program.	Issue the asncmd system command with chgparms autoprun=y . Set the prune_interval parameter to specify how frequently the automatic pruning process occurs.

2. If you want to prune the dynamic control tables once, use one of the following methods:

Method	Description
Replication Center	Use the Prune Capture Control Tables window to prune the tables once. To open the window, click the Capture Control Servers folder in the Operations branch of the object tree, right-click a server in the contents pane, and click Prune Capture .
Initiate pruning once from a running Capture program.	Issue the asncmd system command with the prune parameter.

CD and UOW table pruning

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.

Pruning progress is indicated by the SYNCHPOINT column value in the 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.

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.

Recommendations for pruning other dynamic control tables

You should regularly prune your replication control tables to remove obsolete data and to improve system performance.

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. Noncondensed 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 noncondensed 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 IBMSNAP_APPLYTRAIL and 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

These topics describe methods to prevent and recover from replication failures that can affect your control tables and replication 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 or warmsi option on Linux, UNIX, Windows, or z/OS, the program attempts to retrieve log records based on the restart point in the IBMSNAP_RESTART table. If the Capture program cannot find the log, the Capture warm start fails.

To prevent a cold start of the Capture program, consider the following recommendations.

- **iSeries:** Start the capture program with the RESTART(*YES) parameter. The Capture program continues processing from the point where it was when it ended previously. Retain sufficient DB2 log data or journal receivers on your system and that this data is available to SQL replication.
- 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.
- Make sure that you retain sufficient DB2 log data or journal receivers on your system and that this data is available to SQL replication.

Recovering from I/O errors and connectivity failures on your control tables

If replication loses connectivity to a control table, you can recover the table, for other errors the replication programs will shut down.

About this task

If the Capture program detects an I/O error or connectivity failure, the program issues an appropriate error message and shuts down.

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 IBMSNAP_APPLYTRAIL table and then continues processing.

Procedure

To recover from errors and connectivity failures to your control tables:

1. 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.
2. If the programs shut down, restart the Capture program from the point of failure and restart the Apply program.

Retrieving lost source data

If you lose source you can possibly retrieve it through a recovery point method or a full refresh.

About this task

If a source table is forward recovered to the point of failure, SQL 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.

Procedure

Recover your source data using one of the following methods:

Method	Description
Recovery point mechanism	Develop a mechanism that provides matching recovery points among the various levels of replication.
Full refresh	Use a full refresh as your recovery method of choice

IBMSNAP_CAPMON and IBMSNAP_CAPTRACE table pruning

Your operating parameter values determine pruning of the IBMSNAP_CAPMON and IBMSNAP_CAPTRACE tables.

During each pruning cycle, the Capture program prunes the IBMSNAP_CAPMON and the IBMSNAP_CAPTRACE tables based on the values of the following operational parameters of the Capture program:

- The **monitor_limit** parameter (Linux, UNIX, Windows, z/OS) and the **MONLMT** parameter (iSeries) determine how long rows remain in the IBMSNAP_CAPMON table
- The **trace_limit** parameter (Linux, UNIX, Windows, z/OS) and the **TRCLMT** parameter (iSeries) determine 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.

IBMSNAP_SIGNAL table pruning

Because rows are constantly being added during replication, the IBMSNAP_SIGNAL table is pruned automatically.

The 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.

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 SQL replication.

Note: Deactivate your Apply programs before taking a target table offline to run any utility.

Chapter 14. Detecting and repairing differences between source and target tables

The `asntdiff` and `asntrep` utilities allow you to detect and repair differences between source and target tables in Q replication and SQL replication without manually comparing the tables or performing a load (full refresh) of the target.

About this task

Source and target tables can lose synchronization, for example if a target table is unexpectedly changed by a user or application, or if you experienced an extended network or target system outage.

The `asntdiff` and `asntrep` utilities run independently of the Q Capture, Q Apply, Capture, and Apply programs. They use DB2 SQL to fetch data from the source table and the target table and do not use WebSphere MQ queues. The utilities do not depend on logs, triggers, or isolation level.

Procedure

To detect and repair differences between source and target tables, run the `asntdiff` utility, and then run the `asntrep` utility. The following topics describe these two steps.

- “Table difference utility (`asntdiff`)”
- “Table repair utility (`asntrep`)” on page 186

Table difference utility (`asntdiff`)

The `asntdiff` utility compares all columns in a source table to their corresponding columns in a target table and generates a list of differences between the two tables in the form of a DB2 table.

To use the `asntdiff` utility, you run the `asntdiff` command and specify the name of a Q subscription (Q replication) or subscription set member (SQL replication) that contains the source and target tables that you want to compare.

You can run the `asntdiff` command on Linux, UNIX, Windows, and z/OS operating systems. The command compares tables on Linux, UNIX, Windows, z/OS, or iSeries operating systems. The `asntdiff` command can be used with federated sources and targets if the corresponding columns in the two tables have the same data types.

For Q replication, the target must be a table and not a stored procedure. For SQL replication, the target must be a user table, point-in-time table, replica table, or user-copy table.

When you run the command, you specify an SQL `WHERE` clause that uniquely identifies the Q subscription or subscription set member:

Q replication

The WHERE clause identifies a row in the IBMQREP_SUBS control table at the Q Capture server, based on the value of the SUBNAME column. For example:

```
where="subname = 'my_qsub'"
```

SQL replication

The WHERE clause identifies a row in the IBMSNAP_SUBS_MEMBR table at the Apply control server, based on the value of the SET_NAME column. For example:

```
where="set_name = 'my_set' and source_table='EMPLOYEE'"
```

You might need to use more predicates in the WHERE clause to uniquely identify the subscription set member. For example, you might need to add the APPLY_QUAL, the SOURCE_OWNER, the TARGET_OWNER, or the TARGET_TABLE column from the IBMSNAP_SUBS_MEMBR table to the clause.

Difference table

The asntdiff command creates a difference table in the source database or subsystem for Q replication and SQL replication.

The difference table is named *schema*.ASNTDIFF, where *schema* is the value specified in the DIFF_SCHEMA parameter. If the schema is not specified, it defaults to ASN. You can also use the DIFF parameter to specify a table name.

By default, the difference table is created in the default DB2 user table space. You can specify a different table space by using the DIFF_TABLESPACE parameter. You can also create the difference table and specify a different table space. If you create the difference table, any rows that it contains will be deleted when you use the asntdiff command.

The difference table has two or more columns. One column is named DIFF , with a blank space at the end. The value in the DIFF column is a character that indicates an insert, update, or delete operation followed by a numeric value that indicates which table contains a row with differences. The other columns contain the value of replication key columns. There is one row in the difference table for each unmatched row in the target table.

The difference table uses three identifiers that indicate the operation that is needed to change the target table so that it matches the source table:

D (delete)

Indicates that a row with the key value exists only at the target and not at the source.

U (update)

Indicates that rows with the same key value exist at both the source and target, but at least one non-key column is different at the target.

I (insert)

Indicates that a row with the key value exists only at the source and not at the target.

A value of ? 1 indicates that there is an invalid character in one or more source columns.

A value of ? 2 indicates that there is an invalid character in one or more target columns.

Example

The following list of values is returned by comparing an EMPLOYEE table at the source with a target copy of the same table. The key column for replication is the employee number, EMPNO:

```
DIFF  EMPNO
U 2    000010
I 2    000020
I 2    000040
D 2    000045
I 2    000050
D 2    000055
```

The first row in the example shows that a row with the key value 000010 exists at both the source and target tables, but at least one non-key column at the target has a different value. The next two rows show that rows with the key values 000020 and 000040 exist only at the source. The fourth row shows that a row with the key value 000045 exists only at the target.

The values ? 1 and ? 2 are not shown in the example.

Suppressed delete operations

In Q replication, you can choose to suppress replication of delete operations from the source table. If you do not replicate delete operations, rows that exist in the target table may not exist in the source table. When the SUPPRESS_DELETES value for a Q subscription is Y, the asntdiff utility ignores the rows that are unique to the target and reports no differences. A warning is issued to indicate how many rows were suppressed.

Different data types in sources and targets

The asntdiff utility builds two SELECT SQL statements that are based on the description of a subscription. To obtain the differences between the source and target tables, the utility compares the data that result from executing both statements. The data types of the columns for both SQL statements must be the same.

SQL replication

The utility builds the SQL statement for the source by using the EXPRESSION column in the IBMSNAP_SUBS_COLS table.

Q replication

The data types for both the source and the target must be the same.

Comparing the GRAPHIC data type

Columns with the GRAPHIC data type at the source and target might not match when you use the asntdiff utility to compare the source and target tables. DB2 columns with the GRAPHIC data type have blank padding after the graphic data. This padding might be single-byte or double-byte spaces, depending on the code page that the database was created in. This padding might cause data to not match between the source and the target tables, especially if the source and target tables are in different code pages. This padding applies only to GRAPHIC data types and not other graphic data types such as VARGRAPHIC or LONG VARGRAPHIC.

To compare columns with GRAPHIC data types, you must remove the blank padding in the data before you compare the source and target tables by using the DB2 scalar function `rtrim(<column>)`. This function eliminates the code page differences for single-byte or double-byte spaces and ensures that the `asntdiff` utility compares the GRAPHIC data in a consistent manner.

Predicates

In some cases, differences between source and target tables are intentional, for example, if you use a search condition in Q replication to filter which rows are replicated. The utility will not show differences between source and target tables that are a result of predicates.

SQL replication

The utility uses the `PREDICATES` column in the `IBMSNAP_SUBS_MEMBER` table to select rows from the source tables. The value of the `UOW_CD_PREDICATES` column is ignored (`asntdiff` looks directly at the source table, where the Apply program looks at the CD table).

Q replication

The utility uses the value of the `SEARCH_CONDITION` column in the `IBMQREP_SUBS` table to build the `WHERE` clause for the `SELECT` statement.

When to use the `asntdiff` utility

The best time to use the `asntdiff` utility is when the source and target tables are stable. You might want to run the utility when the Q Capture and Q Apply programs or Capture and Apply programs are idle. For example, you could run the utility when the Q Capture program reached the end of the DB2 recovery log and all changes are applied at the target. If applications are still updating the source, the comparison might not be accurate.

If the replication programs are running, you might need to run the `asntdiff` command more than once to get a complete picture of evolving differences between the source and target tables.

Table repair utility (`asntrep`)

The `asntrep` utility repairs differences between source and target tables on all DB2 servers by deleting, inserting, and updating rows in the target table. The utility runs on Linux, UNIX, or Windows operating systems.

The `asntrep` utility uses the difference table that is generated by the `asntdiff` utility to do the following:

- Delete rows from the target table that have no matching key in the source table
- Insert rows that are in the source table but have no matching key in the target table
- Update target rows that have matching keys in the source but different non-key data

For Q replication, the target must be a table; it cannot be a stored procedure. For SQL replication, the target must be a user table, a point-in-time table, a replica table, or a user-copy table. If you use the `asntrep` utility with a Q subscription for peer-to-peer replication, you must repair all of the copies of a logical table two copies at a time.

To use the `trepair` utility, you run the `asntrep` command after you run the `asntdiff` command. The `asntrep` command copies the difference table from the source database or subsystem to the target, and then uses the copy to repair the target table.

The `asntrep` command does not drop the difference table from the target database or subsystem. You must drop the table manually.

To use the `asntrep` command, you provide the same `WHERE` clause that you used for the `asntdiff` command to identify the Q subscription or subscription set member that contains the source and target tables that you want to synchronize.

During the repair process, referential integrity constraints on the target table are not dropped. An attempt to insert or delete a row from a target table can fail if the insert or delete operation violates a referential integrity constraint. Also, a duplicate source row might be impossible to repair at the target if the target has a unique index.

Chapter 15. Monitoring replication with the Replication Alert Monitor

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 and how to configure and operate monitors for your replication or publishing environment.

- “Replication Alert Monitor”
- “Alert conditions and notifications for the Replication Alert Monitor” on page 192
- “Setting up the Replication Alert Monitor” on page 197
- “Operating the Replication Alert Monitor” on page 204
- “Parameters of the Replication Alert Monitor” on page 207

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 automatically checks the status of replication and notifies you about certain conditions that occur in your 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 configure the Replication Alert Monitor in one of two ways:

One monitor

Typically you use one monitor when you have few replication programs 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.

Multiple monitors

Use additional monitors to monitor many replication programs, prioritize the monitoring of certain programs, or split up the monitoring workload. You create independent monitors to check the servers in your system. These monitors do not communicate with each other, but they each send alerts about the servers. When you set up multiple monitors, the control information for each monitor is stored on the server that it is assigned to monitor. 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 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.

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 where alerts from the

Replication Alert Monitor are sent. Alerts can also be directed to the z/OS console. The ASNMAIL exit routine sends e-mail notifications for the monitor. You can modify this exit routine to put the alerts elsewhere such as in a problem management system.

Contact groups

A contact group is a collection of contacts that receive the same alerts.

z/OS console

You can also specify that alerts are sent to the z/OS console.

The Replication Alert Monitor monitors servers on DB2 for Linux, UNIX, Windows, or z/OS operating systems. You can monitor replication programs whose control tables are at Version 8 architecture or later.

Restrictions

- Some platform-specific restrictions apply for monitoring the status of the Q Capture, Q Apply, Capture, or Apply programs. These restrictions apply to the following alert conditions:
 - QCAPTURE_STATUS
 - QAPPLY_STATUS
 - CAPTURE_STATUS
 - APPLY_STATUS

z/OS A monitor that runs from a z/OS server cannot monitor the status of the replication programs unless the programs are running on the same z/OS subsystem as the monitor.

Linux, UNIX, Windows

If the monitor is running on a distributed platform and you want to monitor the status of the replication programs, the DB2 Administration Server must be running on the same system as the Q Capture, Q Apply, Capture, or Apply program whose status you want to monitor. The DAS is required to monitor status regardless of whether the replication programs are running on a distributed platform or z/OS.

For example, MONITOR1, which runs on the SERVER_LINUX1 server, can monitor the status of a Q Apply program that runs on the SERVER_ZOS1 server if the DB2 Administration Server is installed and running on SERVER_ZOS1. See the *IBM® WebSphere Information Integration Replication Installation and Customization Guide for z/OS* for details.

- **Non-DB2 relational databases:** 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.
- **z/OS:** The Replication Alert Monitor can send e-mail notifications by using an SMTP server, but cannot use the ASNMAIL exit routine to handle notification.
- **iSeries:** To monitor iSeries servers, the Replication Alert Monitor must run on a Linux, UNIX, or Windows server and monitor the iSeries server remotely. You cannot set up Monitor control servers on DB2 UDB for iSeries servers.

Alert conditions and notifications for the Replication Alert Monitor

The following topics contain information about alert conditions and notifications for the Replication Alert Monitor:

- “Alert conditions for the Replication Alert Monitor”
- “E-mail notifications for replication alert conditions” on page 195
- “Sending alerts to the z/OS console” on page 196
- “The ASNMAIL exit routine for sending alerts in replication” on page 196

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 sections 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” on page 193
- “Alert conditions for the Capture program” on page 193
- “Alert conditions for the Apply program” on page 194

Alert conditions for the Q Capture program

Table 11 describes the alert conditions for the Q Capture program.

Table 11. 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.

Table 11. Alert conditions for the Q Capture program (continued)

Alert condition	Description
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 12 describes the alert conditions for the Q Apply program.

Table 12. 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.
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 a row is inserted in the IBMQREP_EXCEPTIONS table because of a conflict or SQL error at a target.
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 13 describes the alert conditions for the Capture program.

Table 13. 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.

Table 13. Alert conditions for the Capture program (continued)

Alert condition	Description
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.
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 14 describes the alert conditions for the Apply program.

Table 14. 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_SUBSFALING	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.

Table 14. Alert Conditions for the Apply program (continued)

Alert condition	Description
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.

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 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 monitor groups all alerts for that server together.

In Q replication, the 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 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 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 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.

Sending alerts to the z/OS console

The Replication Alert Monitor can send alerts to the z/OS console as well as to an e-mail address or pager. The monitor must run on z/OS to write to the z/OS console.

About this task

If you already configured an e-mail server to receive alerts from the monitor and then choose to send alerts to the z/OS console, alerts are sent to both the z/OS console and the e-mail server.

Procedure

To set up the Replication Alert Monitor to send alerts to the z/OS console:

Set the **console** parameter to Y by using one of the following options.

Option	Description
JCL	Specify CONSOLE=Y in the job that will start the monitor.
asnmon command (USS)	From a USS command prompt, issue the asnmon command with the console parameter set to Y. For example: asnmon MONITOR_SERVER=SAMPLE MONITOR_QUALIFIER=MONQUAL CONSOLE=Y
Replication Center	Use the Run Now or Save Command window to edit the operational command that will start the monitor so that the console parameter is set to Y, then start the monitor.

The ASNMAIL exit routine for sending alerts in replication

The ASNMAIL exit routine distributes alerts that notify you about specific conditions in your replication environment.

The ASNMAIL exit routine handles notification. This exit routine takes the following input:

```
asnmail email_server to_address subject alert_message alert_message
```

Table 15 describes the inputs for the ASNMAIL exit routine.

Table 15. Inputs for the ASNMAIL exit routine

Input	Description
<i>email_server</i>	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.
<i>to_address</i>	This is the e-mail address of the contact to be notified.
<i>subject</i>	This is the subject in the notification.
<i>alert_message</i>	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 `\sqlib\samples\repl\` directory contains a sample of the ASNMAIL exit routine. The `asnmail.smp` sample contains the input parameters and directions for using the sample program.

z/OS: The Replication Alert Monitor cannot use the ASNMAIL exit routine to handle notification on z/OS. An SMTP server can be used instead.

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.

About this task

The following topics describe things to consider before setting up the monitor.

- “Authorization requirements for the Replication Alert Monitor” on page 198
- “Memory used by the Replication Alert Monitor”
- “Optional: Binding the Replication Alert Monitor program packages (Linux, UNIX, Windows)” on page 198

Procedure

To set up the monitor:

1. Create control tables for each Monitor control server.
2. Define contact information for the monitor.
3. Create one or more monitors.
4. Select alert conditions.
5. Operate the monitor.
6. Optional: Define suspension periods for the monitor.

Memory used by the Replication Alert Monitor

The Replication Alert Monitor uses memory to store definitions and to keep 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.

Recommendation: Do not set a memory quota for the Replication Alert Monitor. If you need to set one, set it to 3 MB.

Authorization requirements for the Replication Alert Monitor

All user IDs that run a Replication Alert Monitor must have authorization to access the Q Capture server or Q Apply server that you want to monitor. A user ID must also have access to the Monitor control tables on the Monitor control server.

User IDs that run a monitor must have the following authorities and privileges:

- SELECT, UPDATE, INSERT, and DELETE privileges for the Monitor control tables
- SELECT privileges on the Q Capture and Q Apply control tables 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 that is used by the Replication Alert Monitor (Linux, UNIX, and Windows only)

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

The Replication Alert Monitor program is bound automatically on Linux, UNIX, and Windows during execution. You can bind packages manually if you want to specify bind options, schedule binding, or check that all bind processes completed successfully.

Procedure

To bind the Monitor program packages:

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

Windows:

drive:\...\sqllib\bnd

Where *drive*: is the drive where DB2 is installed.

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 database by entering the following command:
`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 the following command:

```
db2 connect to database
```

Where *database* is the server that you want to monitor. 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*.

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.

About this task

The server where you create the monitor control tables is called a Monitor control server.

The Monitor control server can be a DB2 for Linux, UNIX, Windows database or a DB2 UDB for z/OS subsystem. 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 monitor control tables, use one of the following methods:

Method	Description
ASNCLP command-line program	Use the CREATE CONTROL TABLES FOR command. For example: CREATE CONTROL TABLES FOR MONITOR CONTROL SERVER;
Replication Center	Use the Create Monitor Control Tables window. To open the window, right-click the Monitor Control Servers folder and select Create Monitor Control Tables .

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.

About this task

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. You can change contact information after monitors are running.

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 replication administrators that contains the contact information for all your replication administrators. You can also copy contact and group information from one server to another.

Contacts that you create for the Replication Alert Monitor in the Replication Center cannot be used in other DB2 centers such as the Task Center or the Health Center. Contacts created in other DB2 centers cannot be used by the Replication Alert Monitor.

Procedure

To define contact information for the Replication Alert Monitor:

1. Create contacts and contact groups for the monitors on a Monitor control server by using one of the following methods:

Method	Description
ASNCLP command-line program	Use the CREATE CONTACT command. For example: <pre>CREATE CONTACT REPLADMIN EMAIL "repladmin@us.ibm.com" DESCRIPTION "replication administration";</pre>
Replication Center	Use the Create Contact window or Create Contact Group window. 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 .

2. Optional: Copy contact information from one Monitor control server to another 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**.
3. Optional: Use the DELEGATE CONTACTS command in the ASNCLP command-line program to delegate an existing contact to a new contact for a specific period of time. For example:

```
DELEGATE CONTACT REPLADMIN TO PERFORMACE FROM "2007-11-22" TO "2007-12-06"
```

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.

Before you begin

Before you create monitors, you must set up the Replication Alert Monitor.

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:
 - a. 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**.
 - b. 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

See the online help for details. 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**.

Selecting alert conditions for the Replication Alert Monitor

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.

About this task

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.

Procedure

To select alert conditions for the Replication Alert Monitor, use one of the following methods:

Method	Description
ASNCLP command-line program	Use one of the following commands: <ul style="list-style-type: none"> • CREATE ALERT CONDITIONS FOR APPLY • CREATE ALERT CONDITIONS FOR CAPTURE • CREATE ALERT CONDITIONS FOR Q CAPTURE • CREATE ALERT CONDITIONS FOR Q APPLY
Replication Center	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: <ul style="list-style-type: none"> • 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

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.

Changing alert conditions for the Replication Alert Monitor

You can change alert conditions while the monitor is running. You do this by changing the alert conditions and then reinitializing the monitor.

Procedure

To change alert conditions, use one of the following methods:

Method	Description
ASNCLP command-line program	Use one of the following commands: <ul style="list-style-type: none"> ALTER ALERT CONDITIONS FOR APPLY ALTER ALERT CONDITIONS FOR CAPTURE ALTER ALERT CONDITIONS FOR Q CAPTURE ALTER ALERT CONDITIONS FOR Q APPLY
Replication Center	Use the Alert Conditions window for a Q Capture program, Q Apply program, Capture program, Apply program, or subscription set. To open the windows, select a monitor in the object tree, right-click a schema or subscription set in the contents pane, and click Change .

After you change the alert conditions, reinitialize the monitor.

Defining suspension periods for the Alert Monitor

You can define suspension periods for a Replication Alert Monitor program. You can create a repeating suspension (for example, every Sunday morning for two hours) or suspend the monitor for a single time period.

About this task

While the monitor is suspended, it will stop checking Q Capture, Q Apply, Capture control, or Apply control servers for all defined alert conditions. When the suspension period ends, the monitor will resume checking.

To define a suspension that repeats, you create a *suspension template*. If you create a template, you can reuse the template on multiple monitored servers.

If you do not create a template, you can specify a start date and time and end date and time for which monitoring on a server is suspended one time.

All dates and times for monitor suspensions are based on the clock at the system where the monitor is running.

Restrictions

Suspensions and suspension templates can only be defined through the ASNCLP command-line program and cannot be defined or viewed through the Replication Center.

Procedure

To suspend the monitor for a defined period:

1. **Optional:** Use the CREATE MONITOR SUSPENSION TEMPLATE command in the ASNCLP command-line program to create a template to define a repeating suspension.

For example, the following command creates a template that suspends the monitor program from 00:00:00 to 04:00:00 every Sunday:

```
CREATE MONITOR SUSPENSION TEMPLATE SUNDAY START TIME 00:00:00
REPEATS WEEKLY DAY OF WEEK SUNDAY FOR DURATION 4 HOURS
```

2. Use the CREATE MONITOR SUSPENSION command in the ASNCLP command-line program to define a start and end point for a one-time suspension or use a suspension template.

For example, the following command creates a suspension called S1, which uses the template SUNDAY to suspend the monitor control server QSRVR1:

```
CREATE MONITOR SUSPENSION NAME S1 FOR SERVER QSRVR1 STARTING DATE 2006-12-10
USING TEMPLATE SUNDAY ENDING DATE 2007-12-31
```

3. Reinitialize the monitor that you want to suspend by using the asnmcmd reinit command.

You can also use the Reinitialize Monitor window in the Replication Center. To open the window, right-click a monitor qualifier in the contents pane and click **Reinitialize Monitor**.

4. **Optional:** Use one of the following commands in the ASNCLP command-line program to list, change, or drop monitor suspensions or suspension templates:

Command	Description
LIST MONITOR SUSPENSION	Generates a list of suspensions on a monitor control server.
ALTER MONITOR SUSPENSION	Allows you to change the following properties of a suspension: <ul style="list-style-type: none"> • The template that is used • The start or end date for using a template • The start or end date for suspending the monitor program one time
DROP MONITOR SUSPENSION	Deletes a suspension from the monitor control tables.
LIST MONITOR SUSPENSION TEMPLATE	Generates a list of suspension templates on a monitor control server.
ALTER MONITOR SUSPENSION TEMPLATE	Allows you to change the frequency and length of monitor suspensions as defined in a suspension template.
DROP MONITOR SUSPENSION TEMPLATE	Deletes a suspension template from the monitor control tables.

Operating the Replication Alert Monitor

You can start, stop, suspend, reinitialize, and perform other operations on the Replication Alert Monitor. The following topics describe these tasks.

- “Starting monitors” on page 205
- “Reinitializing monitors” on page 205
- “Suspending and resuming a monitor” on page 206
- “Ending a monitor suspension” on page 206
- “Stopping monitors” on page 207
- “Reviewing Monitor program messages” on page 207

Starting monitors

You can use several methods to start a monitor. You can also 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.

Before you begin

- Create monitor control tables and a monitor, which includes selecting contacts and alert conditions.
- Create a password file.
- 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:

Method	Description
asnmon system command	Use this command to start a monitor and optionally specify startup parameters.
Automatic Restart Manager (z/OS)	You can set up the ARM recovery system to start a monitor from the z/OS console or TSO.
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 .
Windows Service Control Manager	You can set up the Windows Service Control Manager to run a monitor.

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.

Method	Description
asnmcmd system command	Use the asnmcmd reinit command to reinitialize a running monitor.
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 .

Suspending and resuming a monitor

You can suspend and resume a monitor when you want to temporarily stop monitoring your replication or publishing environment.

About this task

You might consider suspending and resuming a monitor instead of stopping and restarting it in the following situations:

- You do not have the authority to stop and start a monitor.
- A server in your replication environment is being serviced. For example, if a monitor named MONITOR1 is monitoring SERVER_GREEN, which is a Q capture server, and SERVER_GREEN will be shut down for maintenance between 4 and 7 p.m., you could suspend MONITOR1 at 4 p.m. and resume it at 7 p.m. This would prevent MONITOR1 from issuing a QCAPTURE_STATUS alert condition.

If you suspend the monitor while the Capture, Apply, Q Capture, or Q Apply programs are running, the monitor continues where it left off when you resume it. When a monitor is suspended and then resumed, it will not check for alert conditions or issue alerts for conditions that were met while the monitor was suspended.

Procedure

To suspend and resume a monitor:

1. Suspend the monitor by issuing the `asnmcmd suspend` command. The monitor stops checking for alert conditions.
2. Resume the monitor by issuing the `asnmcmd resume` command. The monitor resumes checking for alert conditions.

Ending a monitor suspension

You can end a monitor suspension before its regularly scheduled expiration time by removing the suspension from the monitor control tables and then reinitializing the monitor.

Procedure

To end a monitor suspension:

1. Use the `DROP MONITOR SUSPENSION` command in the `ASNCLP` command-line program to remove the suspension from the control tables at the monitor server.

For example, the following command removes a suspension named `SUSP1`:

```
DROP MONITOR SUSPENSION NAME SUSP1
```

2. Reinitialize the monitor by using one of the following methods:

Method	Description
asnmcmd command	Use the <code>asnmcmd reinit</code> command to prompt the monitor to read its control tables for the most recent changes. The following command reinitializes the monitor identified by the monitor qualifier <code>myqual</code> at the Monitor control server <code>wsdb</code> : <code>asnmcmd monitor_server=wsdb monitor_qual=myqual reinit</code>

Method	Description
Replication Center	Use the Reinitialize Monitor window. In the contents pane, right-click the monitor qualifier that identifies the monitor that you want to reinitialize and click Reinitialize Monitor .

Note: You can also stop and then start the monitor to prompt it to read its control tables.

Stopping monitors

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.

About this task

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.

Procedure

To stop a monitor, use one of the following methods:

Method	Description
asnmcmd stop command	Use this command to stop a monitor.
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 .
Windows Service Control Manager	Stop the DB2 replication service. The monitor will stop automatically when the replication service stops.

Reviewing Monitor program messages

Use the Monitor Messages window to review the messages that were inserted in the 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.

Parameters of the Replication Alert Monitor

You can determine the behavior of the Replication Alert Monitor by setting values for various parameters. The following topics describe the parameters, methods of changing them, and considerations for which values to set.

- “Default values of Replication Alert Monitor parameters” on page 208

- “Descriptions of the Replication Alert Monitor parameters”
- “Changing runtime parameters for the Replication Alert Monitor” on page 211
- “Specifying how often the Replication Alert Monitor runs” on page 211
- “Specifying notification criteria for operational errors” on page 212
- “Specifying prune intervals for data from the Replication Alert Monitor” on page 212

Default values of Replication Alert Monitor parameters

When you use the replication administration tools to create Monitor control tables, default values are set for the monitor operating parameters.

Table 16 shows the default value for each parameter.

Table 16. Default values for Replication Alert Monitor operating parameters

Operational parameter	Default value
alert_prune_limit	10080 minutes
autoprunce	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

Descriptions of the Replication Alert Monitor parameters

This topic describes the following parameters you can use to operate the Replication Alert Monitor:

- “alert_prune_limit”
- “autoprunce” on page 209
- “email_server” on page 209
- “max_notification_minutes” on page 209
- “max_notifications_per_alert” on page 209
- “monitor_errors” on page 209
- “monitor_interval” on page 210
- “monitor_limit” on page 210
- “monitor_path” on page 210
- “runonce” on page 210
- “trace_limit” on page 210

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 keep 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.

Changing runtime parameters for the Replication Alert Monitor

You can change runtime parameters for the Replication Alert Monitor when you start the monitor or while the monitor is running.

About this task

You set initial parameter values when you create a monitor. These values are stored in the IBMSNAP_MONPARMS control table. When you start the monitor, it reads this control table and uses the parameter values.

You can override the saved values at runtime when you start the monitor or while the monitor is running. Any runtime values that you set will only last for the current run. If the monitor is stopped and restarted, it uses the values saved in the control table.

Procedure

1. Change parameters when you start the monitor. Use one of the following methods.

Method	Description
asnmon system command	Specify one or more parameters and values when you use this command to start the monitor.
Replication Center	Use the Specify Monitor Startup Parameters window. To open the window, right-click a Monitor qualifier in the contents pane that identifies the monitor that you want to start, and click Start Monitor . Then click Parameters on the Start Monitor window.

2. Use the `asnmcmd chgparms system` command to change parameters while the monitor is running. You can change the following parameters:
 - **monitor_interval**
 - **autoprun**
 - **alert_prune_limit**
 - **trace_limit**
 - **max_notifications_per_alert**
 - **max_notifications_minutes**

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.

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:

- 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.
- 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.
- 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.

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, 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.

Specifying prune intervals for data from the Replication Alert Monitor

The Replication Alert Monitor can automatically prune your Monitor control tables. You must decide whether the Monitor will prune the tables automatically, and if so, how the Monitor will prune the tables.

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 **autoprun** 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.

Chapter 16. Replication services (Windows)

You can operate each replication program as a system service on the Windows operating system by using the Windows Service Control Manager (SCM).

The following topics explain replication services for Windows and the tasks that you can accomplish with them.

- “Description of Windows services for replication”
- “Creating a replication service” on page 216
- “Starting a replication service” on page 217
- “Stopping a replication service” on page 217
- “Viewing a list of replication services” on page 217
- “Dropping a replication service” on page 218

Description of Windows services for replication

On the Windows operating system, a replication service is a program that starts and stops the Q Capture, Q Apply, Capture, Apply, or Replication Alert Monitor programs.

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.

The following terms describe naming rules for replication services:

Replication service name

The replication service name uniquely identifies each service and is used to stop and start a service. It has the following format:

DB2.instance.alias.program.qualifier_or_schema

Table 17 describes the inputs for the replication service name.

Table 17. Inputs for the replication service name

Input	Description
<i>instance</i>	The name of the DB2 instance.
<i>alias</i>	The database alias of the Q Capture server, Q Apply server, Capture control server, Apply control server, or Monitor control server.
<i>program</i>	One of the following values: QCAP (for Q Capture program), QAPP (for Q Apply program), CAP (for Capture program), APP (for Apply program), or MON (for Replication Alert Monitor program).
<i>qualifier_or_schema</i>	One of the following identifiers: Q Capture schema, Q Apply schema, Capture schema, Apply qualifier, or Monitor qualifier.

Example: The following service name is for a Q Capture program that has the schema ASN and is working with database DB1 under the instance called INST1:

DB2.INST1.DB1.QCAP.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 QCAPTURE 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.

Creating a replication service

You can create a DB2 replication service to start a Q Capture program, Q Apply program, Capture program, Apply program, and the Replication Alert Monitor program on Windows operating systems.

Before you begin

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.

About this task

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 Q Capture, Q Apply, or Capture control 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.

Procedure

To create a replication service, use the `asnsrct` command.

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.

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.

Starting a replication service

After you create a replication service, you can stop it and start it again.

About this task

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.

Procedure

To start a replication service:

Use one of the following methods.

- The Windows Service Control Manager (SCM)
- net stop command

Stopping a replication service

After you create a replication service, you can stop it and start it again.

About this task

Important: When you stop a replication service, the program associated with that service stops automatically. However, if you stop a program by using a replication system command (asnlqcmd, asnlqccmd, asnlccmd, asnlacmd, or asnlmcmd), the service that started the program continues to run. You must stop it explicitly.

Procedure

To stop a replication service:

Use one of the following methods.

- The Windows Service Control Manager (SCM)
- net stop command

Viewing a list of replication services

You can view a list of all your replication services and their properties by using the asnlist command.

Procedure

To view a list of replication services:

1. Use the asnlist command.
2. Optional: Use the asnlist command with the **details** parameter to view a list of replication services and descriptions of each service.

Dropping a replication service

If you no longer need a replication service you can drop it so that it is removed from the Windows Service Control Manager (SCM).

About this task

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.

Procedure

To drop a service for replication commands, use the `asnsdrop` command.

Chapter 17. 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.

The following topics describe the scheduling options on various operating systems.

Scheduling programs on Linux and UNIX operating systems

You can schedule when to start the replication programs on the Linux and UNIX operating system.

Procedure

To schedule replication programs on Linux and UNIX

Use the **at** command to start a replication program at a specific time. Table 18 shows commands that are used to start the replication programs at 3:00 p.m. on Friday:

Table 18. Scheduling commands for the replication programs (Linux, UNIX)

Replication program	Linux or UNIX command
Capture	at 3pm Friday asncap autoprune=n
Apply	at 3pm Friday asnapply applyqual=myqual
Replication Alert Monitor	at 3pm Friday asnmon monitor_server=db2srv1 monitor_qualifier=mymon

Scheduling programs on Windows operating systems

You can schedule when to start the replication programs on the Windows operating system.

Procedure

If you are not using the Windows Service Control Manager, use the AT command to start the programs at a specific time. Before you enter the AT command, start the Windows Schedule Service. Table 19 shows commands that are used to start the replication programs at 3:00 p.m. on Friday:

Table 19. Scheduling commands for the replication programs (Windows)

Replication program	Windows command
Capture	c:\>at 15:00/interactive"c:\SQLLIB\BIN\db2cmd.exe c:\CAPTURE\asncap.exe"
Apply	c:\>AT 15:00 /interactive "c:\SQLLIB\BIN\db2cmd.exe c:\SQLLIB\BIN\asnapply.exe control_server=cntldb apply_qual=qualid1"

Table 19. Scheduling commands for the replication programs (Windows) (continued)

Replication program	Windows command
Replication Alert Monitor	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

You can schedule when to start the replication programs on the z/OS operating system using two different commands.

Procedure

To schedule programs on the z/OS operating system, use the following methods:

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.
4. Use either the \$TA JES2 command or the AT NetView command to start the Capture program or the Apply program at a specific time. See *MVS/ESA JES2 Commands* for more information about using the \$TA JES2 command. See the *NetView for MVS Command Reference* for more information about using the AT NetView command.

Scheduling programs on the iSeries operating system

You can schedule when to start the replication programs on the iSeries operating system.

Procedure

1. If you want to start the Apply program, issue the ADDJOBSCDE command.
2. If you want to start the Capture program, issue the SBMJOB command. For example:

```
SBMJOB CMD('STRDPRCAP...')SCDDATE(...)SCDTIME(...)
```

Chapter 18. How the SQL replication components communicate

The various replication components run independently of one another, but rely on one another for information that each stores in the replication control tables to communicate with one another.

The administration tools

The Replication Center or ASNCLP command-line program creates SQL scripts that insert the initial information about registered sources, subscription sets, and alert conditions in the control tables.

The Capture program or triggers

The Capture program and the Capture triggers update the control tables to indicate the progress of replication and to coordinate the processing of changes.

The Apply program

The Apply program updates the control tables to indicate the progress of replication and to coordinate the processing of changes.

The Replication Alert Monitor

The Replication Alert Monitor reads the control tables that have been updated by the Capture program, Apply program, and the Capture triggers to understand any problems and progress at a server.

The Replication Center, ASNCLP, the Capture program or triggers, and the Apply program

When you register a table, view, or nickname as a replication source, the Replication Center or ASNCLP command-line program creates an SQL script that stores the information for this source in the replication control table that contains all registration information, the IBMSNAP_REGISTER table. The SQL script generated by the administration tools 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
- The names of the CD tables for the underlying tables in this view (only for registered views, and only if the underlying tables are registered)
- The schema name and name of the internal CCD table (where one exists)
- 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 one another. This table has several more columns for related information.

For iSeries 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 as follows:

- IBMSNAP_SUBS_SET table
- IBMSNAP_SUBS_MEMBR table
- IBMSNAP_SUBS_COLS table
- IBMSNAP_SUBS_STMTS table

If the target tables do not already exist, the SQL script generated by the Replication Center also creates them.

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 also has several more columns for related information.

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 lists describe 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 identify those registered replication sources for which it must capture changes. Having done so, it holds their 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 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

1. 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).
2. 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.
3. The Apply program copies any changes from the CD table to the target table.
4. The Apply program updates the IBMSNAP_SUBS_SET table to record how much data the Apply program copied for each subscription set.
5. The Apply program updates the 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

When the Capture program prunes the CD tables, it uses the information located in the IBMSNAP_PRUNE_SET table to determine which changes were applied, and deletes those changes already replicated 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 lists describe 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

1. 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).
2. When the Apply program processes a subscription set for a non-DB2 relational source, it updates the 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.
3. The Apply program checks the IBMSNAP_REGISTER table to determine whether there are changes that need to be replicated.
4. The Apply program copies the changes from the CCD table to the target table.
5. The Apply program updates the IBMSNAP_SUBS_SET table to record how much data the Apply program copied for each subscription set.
6. The Apply program updates the 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

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 administration tools and the Replication Alert Monitor

When you define an alert condition with contacts who will be notified when the condition occurs, the Replication Center or ASNCLP command-line programs create 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 following control tables are updated:

- IBMSNAP_CONDITIONS table
- IBMSNAP_CONTACTS table
- IBMSNAP_GROUPS table
- IBMSNAP_CONTACTGRP table

The IBMSNAP_CONDITIONS tables 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.

The other tables for the Replication Alert Monitor contain information about who will be notified if the alert condition occurs (either an individual contact, a group of contacts, or the z/OS console), how that contact will be notified (through e-mail or pager), and how often the contact will be notified should the condition persist.

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 steps describe how the Replication Alert Monitor monitors conditions for the Capture or Apply program and notifies contacts when an alert condition occurs:

1. The Replication Alert Monitor reads the alert conditions and the contact for each condition (for a Monitor qualifier) in the 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 IBMSNAP_ALERTS table, and notifies the contacts for that condition.

Chapter 19. Viewing reports about the SQL replication programs

The following topics describe methods you can use to report on and analyze your replication environment. Use the information to check the current status of the replication programs or to review historical data to determine recent messages and throughput or latency statistics.

Checking the 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 status of the replication programs:

Replication Center (UNIX, Windows, z/OS)

Use the Check Status window for Capture or Apply. To open the windows, click the **Capture Control Servers** or **Apply Control Servers** folder in the **Operations** branch of the object tree, right-click a server in the contents pane, and click **Check Status**.

Note: You cannot use the Replication Center to query the status of the Replication Alert Monitor.

Command line (UNIX, Windows, z/OS)

- Use the Capture program `asnccmd` system command, **status** parameter.
- Use the Apply program `asnacmd` system command, **status** parameter.
- Use the Replication Alert Monitor `asnmcmd` system command, **status** parameter.

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 the following four threads:
 - Administration thread
 - Pruning thread
 - Worker thread
 - Serialization thread
- The Apply program has the following two threads:
 - Administration thread
 - Worker thread
- The Replication Alert Monitor program has the following three threads:
 - Administration thread
 - Worker thread
 - Serialization thread

Use the messages you receive to determine if your programs are working correctly. Typically, worker threads, administration threads, and pruning threads are in a working state and are performing the tasks that they were designed to perform.

Serialization threads, global signal handlers, are typically in the waiting state and usually waiting for signals. The pruning thread prunes the CD tables and the following replication control tables.

- IBMSNAP_UOW table
- IBMSNAP_CAPTRACE table
- IBMSNAP_CAPMON table
- IBMSNAP_SIGNAL table

If the messages that you receive indicate that a program is functioning, but you find evidence in your environment to the contrary, you must investigate further. 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, examine the IBMSNAP_APPLYTRAIL table for messages that might explain why the data is not being applied. System resource problems might prevent the program from working properly.

If the messages you receive do not indicate a typical state, you might have to take further action as described in Table 20.

Table 20. Actions suggested to address 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 IBMSNAP_APPLYTRAIL or IBMSNAP_CAPTRACE table for messages that explain why the thread stopped. For example, if you receive a message indicating that the pruning thread stopped, check the IBMSNAP_CAPTRACE table to determine 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.

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 that adjustments could be made to improve performance.

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 that adjustments could be made to improve performance.

Historical data is derived from the following control tables:

- IBMSNAP_APPLYTRAIL

- IBMSNAP_APPLYTRACE
- IBMSNAP_CAPMON
- IBMSNAP_CAPTRACE

The frequency with which you prune these tables affects the reports that you can generate. It is recommended that you retain at least one week of data in these tables for the purpose of examination when you are troubleshooting or evaluating performance.

Table 21 describes the historical data that you can view.

Table 21. 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: <ul style="list-style-type: none"> • 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, <ul style="list-style-type: none"> • 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 can 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 do not specify an interval, you can view raw data from the IBMSNAP_APPLYTRAIL table.

The Replication Center windows show results from information contained in various control tables and log files. The following topics describe how you can use historical data to evaluate your replication operations with the Replication Center.

Reviewing Capture program messages

Use the Capture Messages window to review the messages that were inserted in the IBMSNAP_CAPTRACE table over a specific 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, using the Capture Messages window, you can review all of 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 time period. The Capture program regularly records statistical information in the IBMSNAP_CAPMON table, and during pruning it records pruning statistics in the 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 have the option to specify that the following results display in absolute or average values.

- Number of rows inserted from a log or skipped
- Number of rows pruned from a CD table
- Number of transactions committed
- Use of memory

For example, using 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; 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 approximate the intervening time period between the update of certain data at the source and its capture by the Capture program. This elapsed time provides you with some indication of the currency of the data in CD tables over time. This average latency is derived from information located in the IBMSNAP_CAPMON table, which derives its information from the IBMSNAP_REGISTER table.

The current Capture latency is calculated as the difference between the CURRENT_TIMESTAMP value in the SYNCHTIME column and the global record in the IBMSNAP_REGISTER table:

$(CURRENT_TIMESTAMP) - (SYNCHTIME)$

Table 22. Example of values for calculating current Capture latency

Parameter	Column value
CURRENT_TIMESTAMP	2006-10-20-10:30:25

Table 22. Example of values for calculating current Capture latency (continued)

Parameter	Column value
SYNCHTIME	2006-10-20-10:30:00

For example, using the values in Table 22 on page 230, the current latency is 25 seconds:

10:30:25 - 10:30:00

The Capture latency period changes as time goes by and the history of these changes is stored in the IBMSNAP_CAPMON table. The Replication Center also uses information located in the Capture monitor table to calculate average or historical latency. This formula differs from the formula used to calculate current latency in that it uses the MONITOR_TIME value instead of the CURRENT_TIMESTAMP value to calculate average latency. 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 any messages that were inserted in the IBMSNAP_APPLYTRACE table over a specific period of time. The IBMSNAP_APPLYTRACE table contains rows for significant events such as initialization, warnings, and errors that might be issued by the Apply program.

For example, from the Apply Messages window, you can review all of the error and warning messages that might have been recorded by the Apply program during one week. You can also print or save that 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 IBMSNAP_APPLYTRAIL table. The IBMSNAP_APPLYTRAIL table contains data about the execution of subscription sets, including their status, 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 summaries for failed subscription sets

For example, from the Apply Report window, you can determine whether the Apply program successfully processed subscription sets during the last week. You can view the error messages issued by the Apply program for any subscription sets that could not be replicated. In addition, you can use the Apply Report window in conjunction 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 determine the number of rows replicated and the length of time that replication took.

You can also use the Apply Report window to display all of 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.

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 determine the length of 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.

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 period of time. You can also divide the time period into intervals and display the average latency for each interval.

The Replication Center uses the following formula to calculate the end-to-end latency:

$$(\text{ENDTIME} - \text{LASTRUN}) + (\text{SOURCE_CONN_TIME} - \text{SYNCHTIME})$$

- 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 23. Example of values for calculating end-to-end latency

Parameter	Column value
ENDTIME	2006-10-20-10:01:00
LASTRUN	2006-10-20-10:00:30
SOURCE_CONN_TIME	2006-10-20-10:00:32
SYNCHTIME	2006-10-20-10:00:00

For example, assume a particular subscription set has the values that are shown in Table 23. Using the previous equation, the average end-to-end latency for this subscription set is 62 seconds:

Checking the status of the Capture and Apply journal jobs (iSeries)

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.

Procedure

To check the status of the journal jobs for the Capture and Apply programs:

Use the Work with Subsystem Jobs (WRKSBSJOB) system command as follows:

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. Identify jobs of interest from among those listed as running.

The journal job is named after the journal to which it was assigned. If no job is listed there, use the Work with Submitted Jobs (WRKSBJOB) 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 identify why it failed.

Monitoring the progress of the Capture program (iSeries)

If the Capture program has terminated, you can inspect the IBMSNAP_RESTART table to determine how much progress the Capture program made before termination. 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 sequence number of that journal entry.

About this task

If the Capture program has terminated, you can inspect the IBMSNAP_RESTART table to determine how much progress the Capture program made before termination. 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 sequence number of that journal entry.

Procedure

To determine progress of the Capture program while it is running:

1. Open the CD table for each source table being captured.
2. In the last row of each CD table, note the hex value in the COMMITSEQ column.
3. Identify a row in the IBMSNAP_UOW table with the same COMMITSEQ hex value. If no matching COMMITSEQ value exists in the IBMSNAP_UOW table, repeat the process with the second-to-last row in the CD table. Proceed backward through the CD table until you identify a matching hex value.

4. When you find a matching COMMITSEQ hex value, note the value in the LOGMARKER column of the UOW row. This is the timestamp of the last journal entry processed. 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 printer) to preserve the 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.

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 20. Customizing and running replication SQL scripts for SQL replication

To create control tables, register source tables, and create subscription sets and members, you must run SQL scripts that are generated by the Replication Center and ASNCLP command-line program. You can run the SQL scripts using the Replication Center or you can run them from a DB2 command line. If necessary, you can modify the SQL scripts to meet your needs.

Before you begin

If you run the SQL scripts from a DB2 command line, you must connect to servers manually when you run the SQL script, edit the SQL statements to specify the user ID and password for the server to which you are connecting. 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 ;
```

About this task

You have the option in the ASNCLP and Replication Center to run a generated SQL script immediately or to save the generated SQL script to run later. Even if you choose to run the SQL now, you might also want to save it 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 standalone 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 dbspaces or 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.

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`

Chapter 21. Naming rules for SQL replication objects

The following table lists the limits for names of replication objects.

Table 24. Name limits for replication objects

Object	Name limits
Source and target tables	<p>UNIX, Windows, z/OS: Follow the naming rules for your database management system.</p> <p>iSeries: Names cannot include blanks, asterisks (*), question marks (?), single quotation marks ('), double quotation marks ("), or a slash (/).</p>
Source and target columns	<p>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.)</p>
Subscription set	<p>A subscription-set name can include any characters allowed by DB2 for varying-character (VARCHAR) columns.</p> <p>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.</p>
Capture schema	<p>UNIX, Windows: The Capture schema can be a string of 30 or fewer characters¹.</p> <p>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 characters¹.</p> <p>iSeries: The Capture schema (CAPCTLLIB) can be a string of 10 or fewer alphanumeric characters¹.</p>
Apply qualifier	<p>UNIX, Windows, z/OS: The Apply qualifier can be a string of 18 or fewer characters¹.</p> <p>iSeries: 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 qualifier¹.</p>
Monitor qualifier	<p>UNIX, Windows, z/OS: The Monitor qualifier can be a string of 18 or fewer characters¹.</p>

Table 24. Name limits for replication objects (continued)

Object	Name limits
Note:	
1. For Capture schemas, Apply qualifiers, and Monitor qualifiers, ensure that you use only the following valid characters in the names of these objects:	
<ul style="list-style-type: none">• 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 22. System commands for SQL replication (Linux, UNIX, Windows, z/OS)

This section describes commands for Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS that let you start, operate, modify, and monitor SQL replication programs.

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 iSeries operating systems.

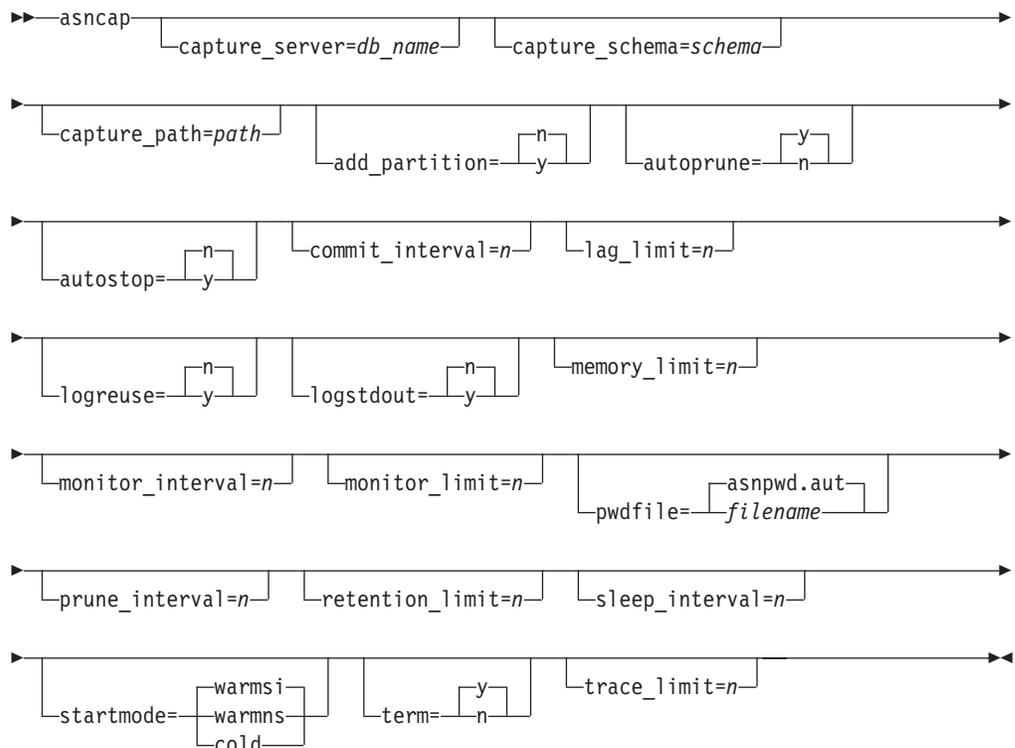
The following topics explain the system commands and their syntax:

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 25 defines the invocation parameters.

Table 25. *asncap* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
<code>capture_server=db_name</code>	<p>Specifies the name of the Capture control server.</p> <p>Linux, UNIX and Windows: If you do not specify a Capture control server, this parameter defaults to the value from the DB2DBDFT environment variable.</p> <p>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.</p>
<code>add_partition=y/n</code>	<p>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.</p> <p>n (default) No new partitions have been added since the last time the Capture program was restarted.</p> <p>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.</p>
<code>capture_schema=schema</code>	<p>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.</p>
<code>capture_path=path</code>	<p>Specifies the location of the work files used by the Capture program. The default is the directory where the <i>asncap</i> command was invoked.</p>
<code>autoprune=y/n</code>	<p>Specifies whether automatic pruning of the rows in the change-data (CD), unit-of-work (UOW), IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_SIGNAL tables is enabled.</p> <p>y (default) The Capture program automatically prunes the eligible rows at the interval specified in the 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.</p> <p>n Automatic pruning is disabled.</p>

Table 25. *asncap* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
autostop =y/n	<p>Specifies whether the Capture program terminates after retrieving all the transactions that were logged before the Capture program started.</p> <p>n (default) The Capture program does not terminate after retrieving the transactions.</p> <p>y The Capture program terminates after retrieving the transactions.</p>
commit_interval =n	<p>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.</p>
lag_limit =n	<p>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.</p>
logreuse =y/n	<p>Specifies whether the Capture program reuses or appends messages to the log file (db2instance.capture_server.capture_schema.CAP.log).</p> <p>n (default) The Capture program appends messages to the log file, even after the Capture program is restarted.</p> <p>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.</p> <p>z/OS: The log file name does not contain the DB2 instance name (capture_server.capture_schema.CAP.log).</p>
logstdout =y/n	<p>Specifies where the Capture program directs the log file messages:</p> <p>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).</p> <p>y The Capture program directs log file messages to both the log file and the standard output (STDOUT).</p>
memory_limit =n	<p>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.</p>
monitor_interval =n	<p>Specifies how frequently (in seconds) the Capture program inserts rows into the IBMSNAP_CAPMON table. The default is 300 seconds (five minutes).</p>
monitor_limit =n	<p>Specifies how long (in minutes) a row can remain in the 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).</p>

Table 25. *asncap* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
pwdfile=filename	<p>Specifies the name of the password file. If you do not specify a password file, the default is asnpwd.aut.</p> <p>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.</p>
prune_interval=n	<p>Specifies how frequently (in seconds) the change-data (CD), unit-of-work (UOW), IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_SIGNAL tables are pruned. This parameter is ignored if you set the autoprune parameter to n. The default is 300 seconds (five minutes).</p>
retention_limit=n	<p>Specifies how long (in minutes) a row can remain in the change-data (CD), unit-of-work (UOW), or 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).</p>
sleep_interval=n	<p>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.</p> <p>z/OS: Specifies the number of seconds that the Capture program sleeps after the buffer returns less than half full.</p>
startmode=mode	<p>Specifies the processing procedure used by the Capture program during a Capture startup.</p> <p>warmsi (default)</p> <p>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.</p> <p>During a warm start, the Capture program leaves the IBMSNAP_CAPTRACE, change-data (CD), unit-of-work (UOW), and IBMSNAP_RESTART tables intact. If errors occur after the Capture program started, the Capture program terminates.</p> <p>warmns</p> <p>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.</p> <p>cold</p> <p>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.</p>

Table 25. *asncap* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
term=y/n	<p>Specifies whether the Capture program terminates if DB2 terminates.</p> <p>y (default) The Capture program terminates if DB2 terminates.</p> <p>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.</p> <p>If DB2 terminates via FORCE or due to abnormal termination, the Capture program terminates even if you set this parameter to n.</p> <p>If you set this parameter to n and start DB2 with restricted access (ACCESS MAINT), the Capture program cannot connect and subsequently terminates.</p>
trace_limit=n	<p>Specifies how long (in minutes) a row can remain in the 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).</p>

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 autoprun=n sleep_interval=10 startmode=warmsi
```

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
```

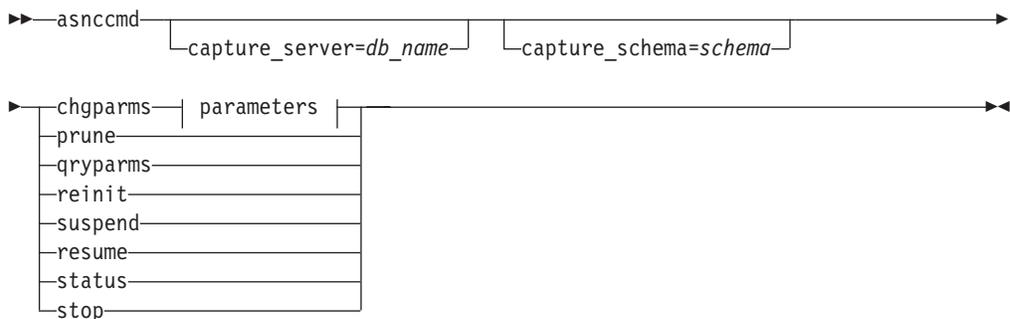
2. Start the Capture program, and specify a Capture path that is located in the new subdirectory that you just created:

```
asncap capture_server=db capture_schema=ASN
      capture_path=/home/db2inst/capture_files startmode=warmsi
```

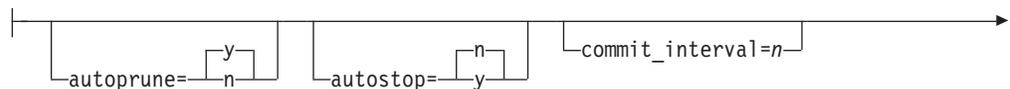
asncmd: Operating Capture

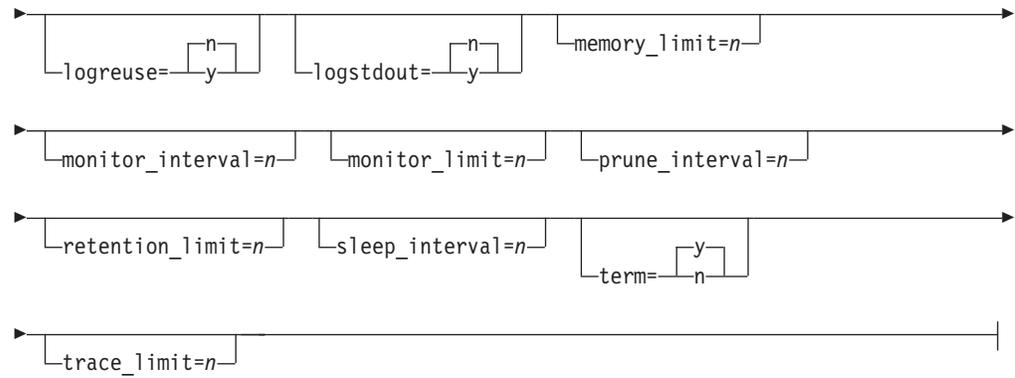
Use the asncmd command to send a command to a running Capture program on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. Run this command at an operating system prompt or in a shell script.

Syntax



Parameters:





Parameters

Table 26 defines the invocation parameters for the asncmd command.

Table 26. Definitions for asncmd invocation parameters

Parameter	Definition
<code>capture_server=y/n</code>	<p>Specifies the name of the Capture control server.</p> <p>Linux, UNIX, Windows: If you do not specify a Capture control server, this parameter defaults to the value from the <i>DB2DBDFT</i> environment variable.</p> <p>z/OS: The name of the database server that connects to the control server. For data sharing, use either the group attach name or a member subsystem name.</p>
<code>capture_schema=schema</code>	<p>Specifies the name of the Capture schema that is used to identify a particular Capture program. The schema name must be 1 to 30 characters in length. The default is ASN.</p>

Table 26. Definitions for `asncmd` invocation parameters (continued)

Parameter	Definition
chgparms	<p>Specify to change one or more of the following operational parameters of a Capture program while it is running:</p> <ul style="list-style-type: none"> • autostop • commit_interval • logreuse • logstdout • memory_limit • monitor_interval • monitor_limit • prune_interval • retention_limit • signal_limit • sleep_interval • term • trace_limit <p>You can specify multiple parameters in one asncmd chgparms command, and you can change these parameter values as often as you want. The changes temporarily override the values in the <code>IBMSNAP_CAPPARMS</code> table, but they are not written to the table. When you stop and restart the Capture program, it uses the values in <code>IBMSNAP_CAPPARMS</code>. “<code>asncap: Starting Capture</code>” on page 239 includes descriptions of the parameters that you can override with this command.</p>
prune	<p>Specify this parameter if you want to prune the change-data (CD), unit-of-work (UOW), <code>IBMSNAP_CAPMON</code>, <code>IBMSNAP_CAPTRACE</code>, and <code>IBMSNAP_SIGNAL</code> tables once. The Capture program issues a message when the command is successfully queued.</p>
qryparms	<p>Specify if you want the current operational parameter values written to the standard output (stdout).</p>
reinit	<p>Specify to have the Capture program obtain newly added replication sources from the <code>IBMSNAP_REGISTER</code> table. For example, use this parameter if you add a new replication source or if you use the <code>ALTER ADD</code> statement to add a column to a replication source and change-data (CD) table while the Capture program is running.</p>
suspend	<p>Specify to relinquish operating system resources to operational transactions during peak periods without destroying the Capture program environment.</p> <p>Attention: Do not suspend Capture to cancel a replication source. Instead, stop the Capture program.</p>
resume	<p>Specify to have a suspended Capture program resume capturing data.</p>
status	<p>Specify to receive messages that indicate the state of each Capture thread (administration, pruning, serialization, and worker).</p>
stop	<p>Specify to stop the Capture program in an orderly way and commit the log records processed up to that point.</p>

Examples for asncmd

The following examples illustrate how to use the asncmd command.

Example 1

To enable a running Capture program to recognize newly added replication sources:

```
asncmd capture_server=db capture_schema=ASN reinit
```

Example 2

To prune the CD, UOW, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_SIGNAL tables once:

```
asncmd capture_server=db capture_schema=ASN prune
```

Example 3

To receive messages about the state of each Capture thread:

```
asncmd capture_server=db capture_schema=ASN status
```

Example 4

To send the current operational values of a Capture program to the standard output:

```
asncmd capture_server=db capture_schema=ASN qryparms
```

Example 5

To disable the automatic pruning in a running Capture program:

```
asncmd capture_server=db capture_schema=ASN chgparms autoprun=n
```

Example 6

To stop a running Capture program:

```
asncmd capture_server=db capture_schema=ASN stop
```

asnapply: Starting Apply

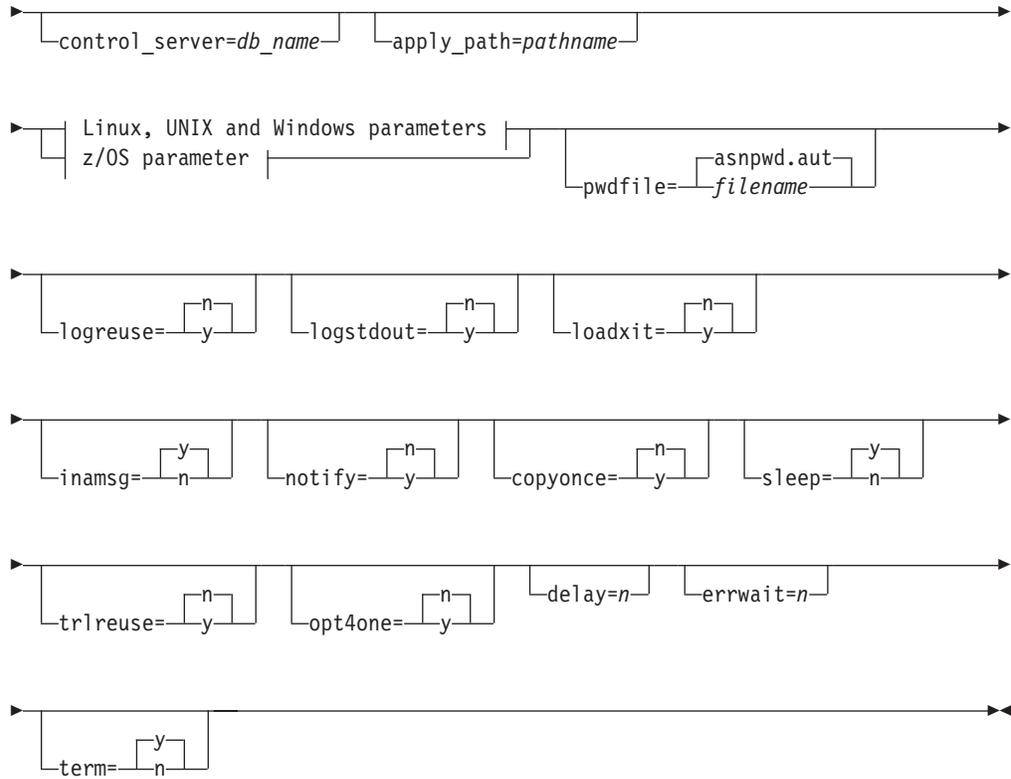
Use the asnapply command to start the Apply program on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. Run this command at an operating system prompt or in a shell script.

After you start the Apply program, it runs continuously until:

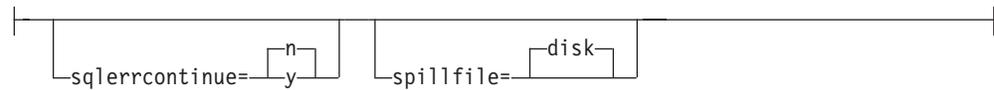
- You stop it in an orderly way.
- You cancel it.
- An unexpected error or failure occurs.

Syntax

```
▶▶—asnapply—apply_qual=apply_qualifier—db2_subsystem=name————→(1)
```



Linux, UNIX, Windows and z/OS parameters:



z/OS parameter:



Notes:

- 1 Use the **db2_subsystem** parameter only on z/OS operating systems.

Parameters

Table 27 defines the invocation parameters.

Table 27. *asnapply* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
apply_qual = <i>apply_qualifier</i>	<p>Specifies the Apply qualifier that the Apply program uses to identify the subscriptions sets to be served.</p> <p>The value that you enter must match the value of the APPLY_QUAL column in the IBMSNAP_SUBS_SET table. The Apply qualifier name is case sensitive and can be a maximum of 18 characters.</p>
db2_subsystem = <i>name</i>	<p>z/OS: 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.</p>
control_server = <i>db_name</i>	<p>Specifies the name of the Apply control server on which the subscription definitions and Apply program control tables reside.</p> <p>Linux, UNIX and Windows: If you do not specify an Apply control server, this parameter defaults to the value from the DB2DBDFT environment variable.</p> <p>z/OS: Specifies the location name of the Apply control server.</p>
apply_path = <i>pathname</i>	<p>Specifies the location of the work files used by the Apply program. The default is the directory where the asnapply command was invoked.</p>
pwdfile = <i>filename</i>	<p>Specifies the name of the password file. If you do not specify a password file, the default is <code>asnpwd.aut</code>.</p> <p>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.</p>
logreuse = <i>y/n</i>	<p>Specifies whether the Apply program reuses or appends messages to the log file (<code>db2instance.control_server.apply_qualifier.APP.log</code>).</p> <p>n (default) The Apply program appends messages to the log file, even after the Apply program is restarted.</p> <p>y The Apply program reuses the log file by deleting it and then re-creating it when the Apply program is restarted.</p> <p>z/OS: The log file name does not contain the DB2 instance name (<code>control_server.apply_qualifier.APP.log</code>).</p>

Table 27. *asnapply* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
logstdout=y/n	<p>Specifies where the Apply program directs the log file messages:</p> <p>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).</p> <p>y The Apply program directs log file messages to both the log file and the standard output (STDOUT).</p>
loadxit=y/n	<p>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.</p> <p>n (default) The Apply program does not invoke ASNLOAD.</p> <p>y The Apply program invokes ASNLOAD.</p>
inamsg=y/n	<p>Specifies whether the Apply program issues a message when the Apply program is inactive.</p> <p>y (default) The Apply program issues a message when inactive.</p> <p>n The Apply program does not issue a message when inactive.</p>
notify=y/n	<p>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.</p> <p>n (default) The Apply program does not invoke ASNDONE.</p> <p>y The Apply program invokes ASNDONE.</p>
copyoncecopyonce	<p>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:</p> <ul style="list-style-type: none"> • (ACTIVATE > 0) in the 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. <p>The MAX_SYNCH_MINUTES limit from the subscription sets table and the END_OF_PERIOD timestamp from the IBMSNAP_SUBS_EVENT table are honored if specified.</p> <p>n (default) The Apply program does not execute one copy cycle for each eligible subscription set.</p> <p>y The Apply program executes one copy cycle for each eligible subscription set.</p>

Table 27. *asnapply* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
sleep=y/n	<p>Specifies how the Apply program is to proceed if no new subscription sets are eligible for processing.</p> <p>y (default) The Apply program goes to sleep.</p> <p>n The Apply program stops.</p>
trlreuse=y/n	<p>Specifies whether the Apply program empties the IBMSNAP_APPLYTRAIL table when the Apply program starts.</p> <p>n (default) The Apply program appends entries to the IBMSNAP_APPLYTRAIL table. The Apply program does not empty the table.</p> <p>y The Apply program empties the IBMSNAP_APPLYTRAIL table during program startup.</p>
opt4one=y/n	<p>Specifies whether the performance of the Apply program is optimized if only one subscription set is defined for the Apply program.</p> <p>n (default) The performance of the Apply program is not optimized for one subscription set.</p> <p>y The performance of the Apply program is optimized for one subscription set.</p> <p>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.</p>
delay=n	<p>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.</p>
errwait=n	<p>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).</p> <p>Important: Do not specify too small a number, because the Apply program runs almost continuously and generates many rows in the IBMSNAP_APPLYTRAIL table.</p>

Table 27. *asnapply* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
term=y/n	<p>Specifies how the status of DB2 affects the operation of the Apply program.</p> <p>y (default) The Apply program terminates if DB2 terminates.</p> <p>n The Apply program waits for DB2 to start, if DB2 is not active.</p> <p>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.</p> <p>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.</p>
sqlerrcontinue=y/n	<p>Specifies whether the Apply program continues processing when it encounters certain SQL errors.</p> <p>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.</p> <p>n (default) The Apply program does not check the SQLSTATE file.</p> <p>y The Apply program checks the SQLSTATE file during processing.</p>
spillfile=filetype	<p>Specifies where the fetched answer set is stored.</p> <p>Linux, UNIX and Windows, valid values are:</p> <p>disk (default) A disk file.</p> <p>z/OS, valid values are:</p> <p>mem (default) A memory file. The Apply program fails if there is insufficient memory for the answer set.</p> <p>disk A disk file.</p>

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.

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

```
▶▶—asnacmd—apply_qual=apply_qualifier—┐
└────────────────────────────────────────┘
▶┐status──────────────────────────────────▶▶
└──stop──────────────────────────────────▶▶
```

Parameters

Table 28 defines the invocation parameters.

Table 28. asnacmd invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
<code>apply_qual=<i>apply_qualifier</i></code>	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 IBMSNAP_SUBS_SET table. The Apply qualifier name is case sensitive and can be a maximum of 18 characters.

Table 28. *asnacmd* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
<code>control_server=db_name</code>	Specifies the name of the Apply control server on which the subscription definitions and Apply control tables reside. Linux, UNIX and Windows: If you do not specify an Apply control server, this parameter defaults to the value from the DB2DBDFT environment variable. z/OS: The control server parameter is the name of the database server that connects to the control server.
<code>status</code>	Specify to receive messages that indicate the state of each thread (administration and worker) in Apply.
<code>stop</code>	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:

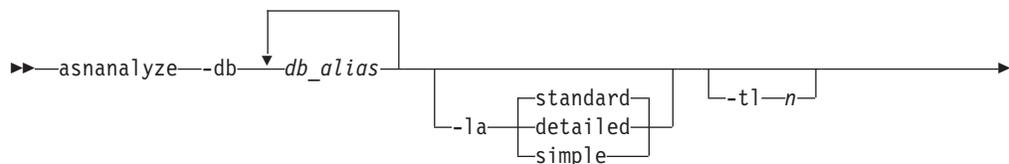
```
asnacmd apply_qual=AQ1 control_server=dbx stop
```

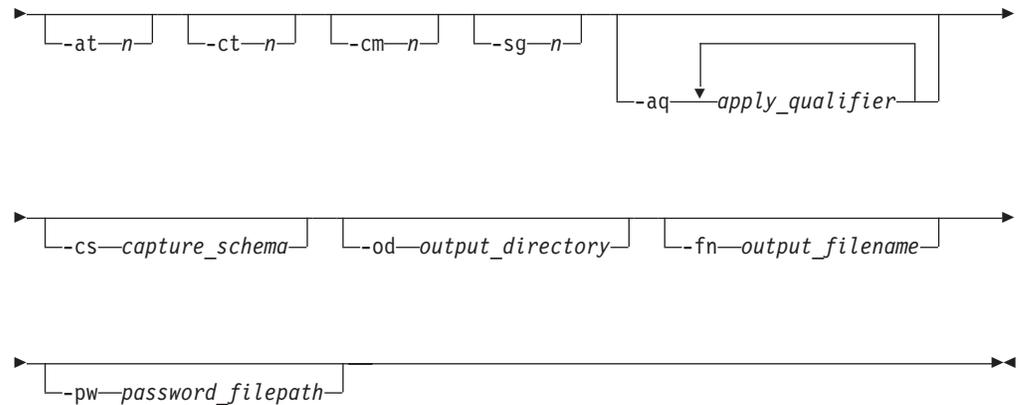
asnanalyze: Operating the Analyzer

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 iSeries 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 29 defines the invocation parameters.

Table 29. *asnanalyze* invocation parameter definitions for Linux, UNIX and Windows operating systems

Parameter	Definition
<code>-db db_alias</code>	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.
<code>-la level_of_analysis</code>	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: <ul style="list-style-type: none"> • 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 IBMSNAP_SUBS_COLS table.
<code>-tl n</code>	Specifies the date range (0 to 30 days) of entries to be retrieved from the IBMSNAP_APPLYTRAIL table. The default is 3 days.
<code>-at n</code>	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.

Table 29. *asnanalyze* invocation parameter definitions for Linux, UNIX and Windows operating systems (continued)

Parameter	Definition
-ct <i>n</i>	Specifies the date range (0 to 30 days) of entries to be retrieved from the IBMSNAP_CAPTRACE table. The default is 3 days.
-cm <i>n</i>	Specifies the date range (0 to 30 days) of entries to be retrieved from the IBMSNAP_CAPMON table. The default is 3 days.
-sg <i>n</i>	Specifies the date range (0 to 30 days) of entries to be retrieved from the IBMSNAP_SIGNAL table. The default is 3 days.
-aq <i>apply_qualifier</i>	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 <i>capture_schema</i>	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 <i>output_directory</i>	Specifies the directory in which you want to store the Analyzer report. The default is the current directory.
-fn <i>output_filename</i>	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 <i>asnanalyze.htm</i> .
-pw <i>password_filepath</i>	Specifies the name and path of the password file. If you do not specify this parameter, the Analyzer checks the current directory for the <i>asnpwd.aut</i> 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 -t1 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 -t1 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
```

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:

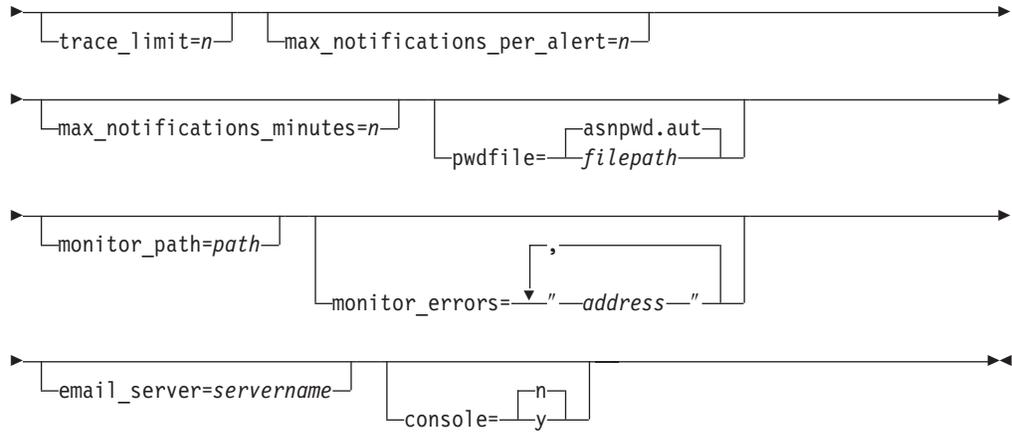
- The status of Q Capture, Q Apply, Capture, and Apply programs
- Error messages written to the control tables
- Threshold values

Syntax

```

▶▶▶ asnmon [monitor_server=server] monitor_qual=mon_qual
      [monitor_interval=n] [runonce=[y|n]] [autoprune=[y|n]]
      [logreuse=[y|n]] [logstdout=[y|n]] [alert_prune_limit=n]

```



Parameters

Table 30 defines the invocation parameters for the asnmon command.

Table 30. asnmon invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
monitor_server=server	<p>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.</p> <p>Linux, UNIX, Windows: If you do not specify a Monitor control server, this parameter defaults to the value from the DB2DBDFT environment variable.</p> <p>z/OS: The default is DSN.</p>
monitor_qual=mon_qual	<p>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.</p> <p>You must specify a monitor qualifier. The monitor qualifier name is case sensitive and can be a maximum of 18 characters.</p>
monitor_interval=n	<p>Specifies how frequently (in seconds) the Replication Alert Monitor program runs for this monitor qualifier. The default is 300 seconds (five minutes).</p> <p>This parameter is ignored by the Replication Alert Monitor if you set the runonce parameter to y.</p> <p>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.</p>

Table 30. *asnmon* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
runonce=y/n	<p>Specifies whether the Replication Alert Monitor program runs only one time for this monitor qualifier.</p> <p>n (default) The Replication Alert Monitor program runs at the frequency indicated by the monitor_interval parameter.</p> <p>y The Replication Alert Monitor program runs only one monitor cycle.</p> <p>If you set the runonce parameter to <i>y</i>, the monitor_interval parameter is ignored by the Replication Alert Monitor.</p>
autoprune=y/n	<p>Specifies whether automatic pruning of the rows in the Replication Alert Monitor alerts (IBMSNAP_ALERTS) table is enabled.</p> <p>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.</p> <p>n Automatic pruning is disabled.</p>
logreuse=y/n	<p>Specifies whether the Replication Alert Monitor program reuses or appends messages to its diagnostic log file (db2instance.monitor_server.mon_qual.MON.log).</p> <p>n (default) The Replication Alert Monitor program appends messages to the log file.</p> <p>y The Replication Alert Monitor program reuses the log file by deleting it and then recreating it when the Replication Alert Monitor program is restarted.</p>
logstdout=y/n	<p>Specifies where messages are sent by the Replication Alert Monitor program.</p> <p>n (default) The Replication Alert Monitor program sends messages to the log file only.</p> <p>y The Replication Alert Monitor program sends messages to both the log file and the standard output (stdout).</p>
alert_prune_limit=n	<p>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).</p>
trace_limit=n	<p>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).</p>

Table 30. *asnmon* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
max_notifications_per_alert = <i>n</i>	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 = <i>n</i>	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 = <i>filepath</i>	Specifies the fully qualified name of the password file. You define this file using the <code>asnpwd</code> command. The default file name is <code>asnpwd.aut</code> .
monitor_path = <i>path</i>	Specifies the location of the log files used by the Replication Alert Monitor program. The default is the directory where the <code>asnmon</code> command was invoked.
monitor_errors = <i>address</i>	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 = <i>servername</i>	Specifies the e-mail server address. Enter this parameter <i>only</i> if you use the <code>ASNMAIL</code> exit routine with SMTP (Simple Mail Transfer Protocol).
console = <i>y/n</i>	Specifies whether the Replication Alert Monitor program sends alert notifications to the z/OS console. If you set this parameter to Y (yes) and an e-mail server was already configured, alerts are sent to both the z/OS console and the e-mail server. n (default) The Replication Alert Monitor program does not send alert notifications to the z/OS console. y The Replication Alert Monitor program sends alert notifications to the z/OS console.

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).

asnmcmd: Working with a running Replication Alert Monitor

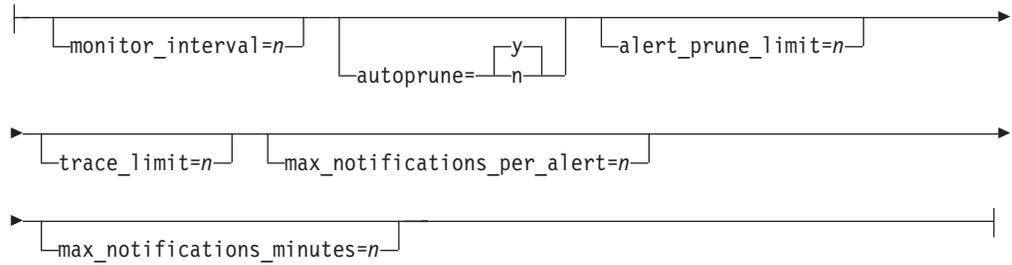
Use asnmcmd to send commands to a running Replication Alert Monitor on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. Run this command at an operating system prompt or in a shell script.

Syntax

```
▶▶ asnmcmd [monitor_server=server] monitor_qual=mon_qual ▶▶
```



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	<p>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.</p> <p>Linux, UNIX, Windows: If you do not specify a Monitor control server, this parameter defaults to the value from the DB2DBDFT environment variable.</p> <p>z/OS: The default is DSN.</p>
monitor_qual=mon_qual	<p>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.</p> <p>You must specify a monitor qualifier. The monitor qualifier name is case sensitive and can be a maximum of 18 characters.</p>

Table 31. *asnmcmd* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
chgparms	<p>Specify to change one or more of the following operational parameters of the Replication Alert Monitor while it is running:</p> <ul style="list-style-type: none"> • monitor_interval • autoprun • alert_prune_limit • trace_limit • max_notifications_per_alert • max_notifications_minutes <p>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. “<i>asnmcmd: Starting a Replication Alert Monitor</i>” on page 257 includes descriptions of the parameters that you can override with this subcommand.</p> <p>Important: The parameter that you are changing must immediately follow the chgparms subcommand.</p>
reinit	<p>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.</p>
status	<p>Specify to receive messages that indicate the state of each thread (administration, serialization, and worker) in the Replication Alert Monitor.</p>
qryparms	<p>Specify if you want the current operational parameter values for the Replication Alert Monitor written to the standard output (stdout).</p>
suspend	<p>Specify if you want the Replication Alert Monitor to stop checking conditions on servers temporarily until you issue the resume command.</p>
resume	<p>Specify if the Replication Alert Monitor has been suspended and you want the Monitor program to begin checking conditions on servers again.</p>
stop	<p>Specify to stop the Replication Alert Monitor in an orderly way.</p>

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
```

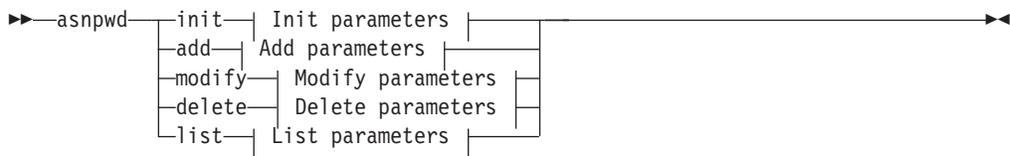
asnpwd: Creating and maintaining password files

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:



Table 32. *asnpwd* invocation parameter definitions for Linux, UNIX, and Windows operating systems (continued)

Parameter	Definition
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 <i>filepath_name</i>	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:\sql\lib\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 <i>asnpwd.aut</i> and the default file path is the current directory.
alias <i>db_alias</i>	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 <i>userid</i>	Specifies the user ID that has access to the database.
password <i>password</i>	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.

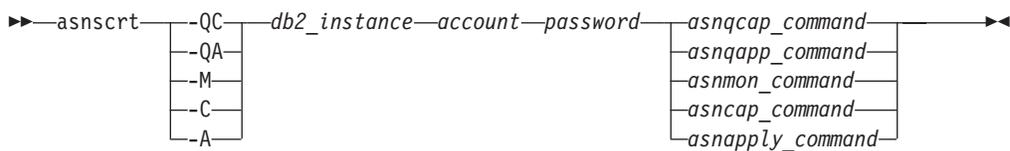
- If it was not initialized using the **encrypt all** parameter, the following details are listed:

```
asnpwd LIST USING pass1.aut
Alias: SAMPLE ID: chglocalid
Number of Entries: 1
```

asnsrct: Creating a replication service

Use the asnsrct command to create a replication service in the Windows Service Control Manager (SCM) and invoke the asnqcap, asnqapp, asnmon, asncap, and asnaply commands. Run the asnsrct command on the Windows operating system.

Syntax



Parameters

Table 33 defines the invocation parameters for the asnsrct command.

Table 33. asnsrct 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 asnsrct command does not validate the syntax of the asnqcap parameters that you enter.

Table 33. *asnscri* invocation parameter definitions for Windows operating systems (continued)

Parameter	Definition
asnsqapp_command	<p>Specifies the complete <i>asnsqapp</i> command to start a Q apply program. Use the documented <i>asnsqapp</i> command syntax with the appropriate <i>asnsqapp</i> parameters.</p> <p>Important: If the <code>DB2PATH</code> environment variable is not defined, you must specify the location for the work files by including the apply_path parameter with the <i>asnsqapp</i> command. If the <code>DB2PATH</code> variable is defined and you specify an apply_path, the apply_path parameter overrides the <code>DB2PATH</code> variable. The <i>asnscri</i> command does not validate the syntax of the <i>asnsqapp</i> parameters that you enter.</p>
asnsmon_command	<p>Specifies the complete <i>asnsmon</i> command to start a Replication Alert Monitor program. Use the documented <i>asnsmon</i> command syntax with the appropriate <i>asnsmon</i> parameters.</p> <p>Important: If the <code>DB2PATH</code> environment variable is not defined, you must specify a location for the log files by including the monitor_path parameter with the <i>asnsmon</i> command. If the <code>DB2PATH</code> variable is defined and you specify a monitor_path, the monitor_path parameter overrides the <code>DB2PATH</code> variable.</p> <p>The <i>asnscri</i> command does not validate the syntax of the <i>asnsmon</i> parameters that you enter.</p>
asnsncap_command	<p>Specifies the complete <i>asnsncap</i> command to start a Capture program. Use the documented <i>asnsncap</i> command syntax with the appropriate <i>asnsncap</i> parameters.</p> <p>Important: If the <code>DB2PATH</code> environment variable is not defined, you must specify a location for the work files by including the capture_path parameter with the <i>asnsncap</i> command. If the <code>DB2PATH</code> variable is defined and you specify a capture_path, the capture_path parameter overrides the <code>DB2PATH</code> variable.</p> <p>The <i>asnscri</i> command does not validate the syntax of the <i>asnsncap</i> parameters that you enter.</p>
asnsapply_command	<p>Specifies the complete <i>asnsapply</i> command to start an Apply program. Use the documented <i>asnsapply</i> command syntax with the appropriate <i>asnsapply</i> parameters.</p> <p>Important: If the <code>DB2PATH</code> environment variable is not defined, you must specify the location for the work files by including the apply_path parameter with the <i>asnsapply</i> command. If the <code>DB2PATH</code> variable is defined and you specify an apply_path, the apply_path parameter overrides the <code>DB2PATH</code> variable.</p> <p>The <i>asnscri</i> command does not validate the syntax of the <i>asnsapply</i> parameters that you enter.</p>

Examples for asnscri

The following examples illustrate how to use the asnscri command.

Example 1

To create a DB2 replication service that invokes a Q capture program under a DB2 instance called inst1:

```
asnscri -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:

```
asnscri -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:

```
asnscri -C inst1 .\joesmith password asncap capture_server=sampled
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:

```
asnscri -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:

```
asnscri -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**:

```
asnscri -C inst4 .\joesmith password X:\sqllib\bin\asncap capture_server=scdb
capture_schema=ASN capture_path=X:\logfiles
```

asnsdrop: Dropping replication services

Use the asnsdrop command to drop replication services from the Windows Service Control Manager (SCM) on the Windows operating system. (You create a replication service using the asnscri command.)

Syntax

```
▶▶ asnsdrop [service_name | ALL] ▶▶
```

Parameters

Table 34 defines the invocation parameters for the asnsdrop command.

Table 34. 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 (with embedded blanks), use double quotation marks around the service name:

```
asnsdrop "DB2.SAMPLEDB.SAMPLEDB.CAP.A S N"
```

Example 3

To drop all DB2 replication services:

```
asnsdrop ALL
```

asnslist: Listing replication services

Use the asnslist command to list 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

```
▶▶ asnslist [DETAILS] ▶▶
```

Parameters

Table 35 defines the invocation parameter for the `asnslist` command.

Table 35. *asnslist* invocation parameter definition for Windows operating systems

Parameter	Definition
<code>details</code>	Specifies that you want to list detailed data about all DB2 replication services on a system.

Examples for `asnslist`

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
```

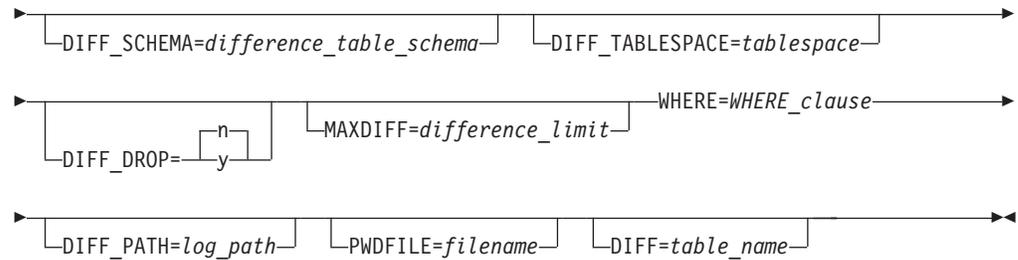
asntdiff: Comparing data in source and target tables

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 tables on Linux, UNIX, Windows, z/OS, and iSeries operating systems.

Syntax

```
▶▶ asntdiff—DB=server—DB2_SUBSYSTEM=subsystem—SCHEMA=schema—▶▶
```



Parameters

Table 36 defines the invocation parameters for the `asntdiff` command.

Table 36. `asntdiff` invocation parameter definitions for Linux, UNIX, and Windows operating systems

Parameter	Definition
<code>DB=server</code>	<p>Specifies the DB2 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:</p> <p>Q replication The value is the name of the Q Capture server, which contains the <code>IBMQREP_SUBS</code> table.</p> <p>SQL replication The value is the name of the Apply control server, which contains the <code>IBMSNAP_SUBS_MEMBR</code> table.</p> <p>z/OS: The value of this parameter is a location name.</p>
<code>DB2_SUBSYSTEM=subsystem</code>	On z/OS, specifies the name of the subsystem where the control tables are created.
<code>SCHEMA=schema</code>	Specifies the schema of the Q Capture control tables for Q replication, or the schema of the Apply control tables for SQL replication. The default is <code>ASN</code> .
<code>DIFF_SCHEMA=difference_table_schema</code>	Specifies the schema that qualifies the difference table. The default is <code>ASN</code> .
<code>DIFF_TABLESPACE=tablespace</code>	<p>Specifies the table space where the difference table will be placed. If this parameter is not specified, the table will be created in the default table space in the database or subsystem where the <code>asntdiff</code> command was run.</p> <p>z/OS: This is a two-part name, <code>dbname.tablespace</code>, where <code>dbname</code> is the logical database name and <code>tablespace</code> is the table space name.</p>

Table 36. *asntdiff* invocation parameter definitions for Linux, UNIX, and Windows operating systems (continued)

Parameter	Definition
DIFF_DROP=y/n	<p>Specifies whether an existing difference table will be dropped and recreated before it is used to record differences. If the table does not exist, the <i>asntdiff</i> command creates it.</p> <p>n (default) The difference table will be used as is and the existing rows will be deleted.</p> <p>y The difference table will be dropped and recreated.</p>
MAXDIFF=<i>difference_limit</i>	<p>Specifies the maximum number of differences that you want the <i>asntdiff</i> command to process before it stops. The default value is 10000.</p>
WHERE=<i>WHERE_clause</i>	<p>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:</p> <p>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.</p> <p>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.</p>
DIFF_PATH=<i>log_path</i>	<p>Specifies the location where you want the <i>asntdiff</i> utility to write its log. The default value is the directory where you ran the command. The value must be an absolute path name. Use double quotation marks (") to preserve case.</p>
PWDFILE=<i>filename</i>	<p>Specifies the name of the password file that is used to connect to databases. If you do not specify a password file, the default value is <i>asnpwd.aut</i> (the name of the password file that is created by the <i>asnpwd</i> command). The <i>asntdiff</i> command searches for the password file in the directory that is specified by the DIFF_PATH parameter. If no value for the DIFF_PATH parameter is specified, the command searches for the password file in the directory where the command was run.</p>

Table 36. *asntdiff* invocation parameter definitions for Linux, UNIX, and Windows operating systems (continued)

Parameter	Definition
<code>DIFF=table_name</code>	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 <code>ASN.ASNTDIFF</code> .

Examples for *asntdiff*

The following examples illustrate how to use the *asntdiff* command.

z/OS: See the *asntdiff* sample program for sample JCL to run 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="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`:

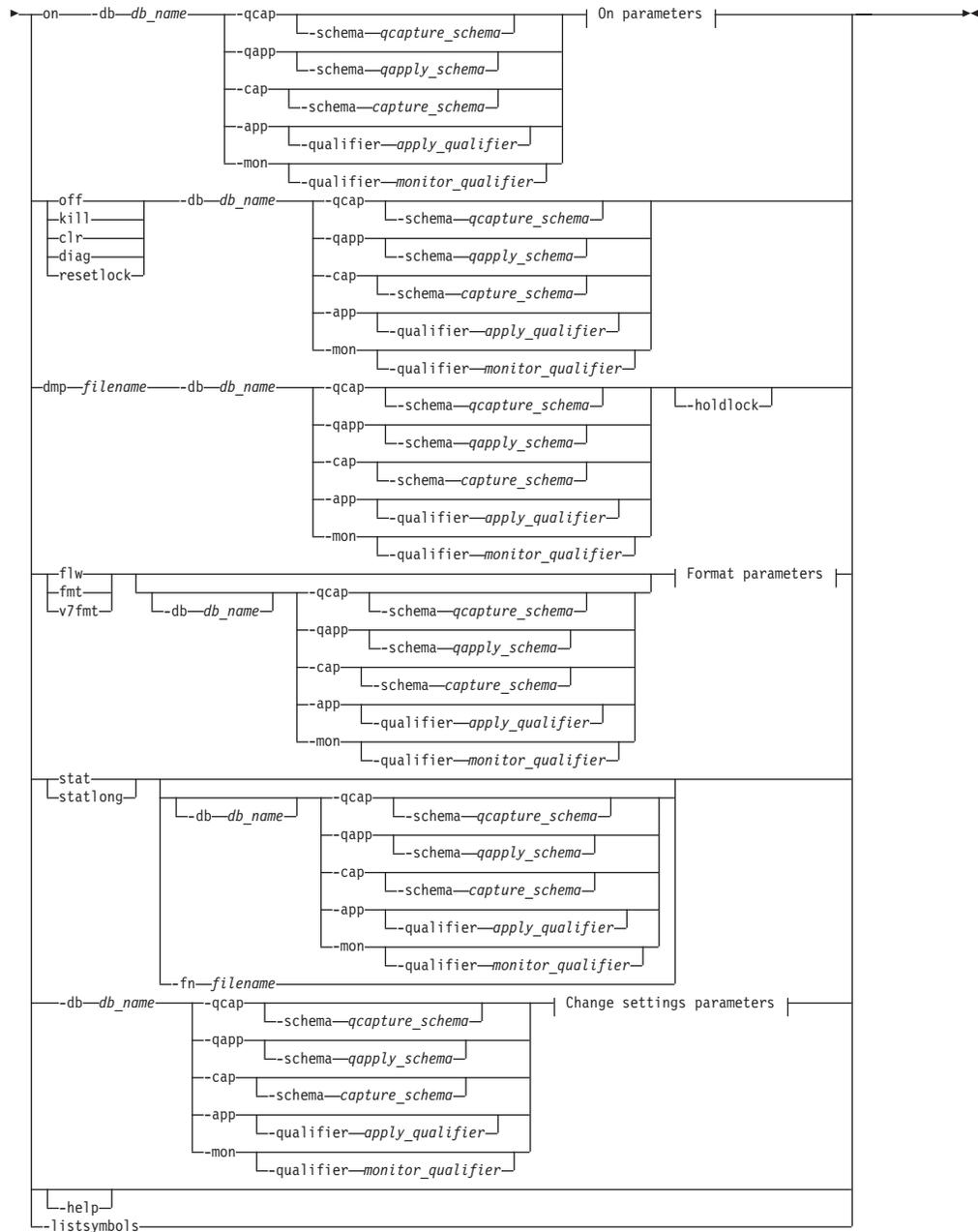
```
asntdiff DB=apply_db schema=asn where="set_name = 'my_set'
and target_table = 'trg_table'" diff=diff_table
```

asntrc: Operating the replication trace facility

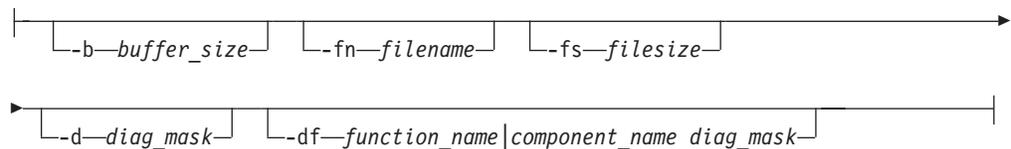
Use the *asntrc* command to run the trace facility on Linux, UNIX, Windows, and UNIX System Services (USS) on z/OS. The trace facility logs program flow information from Q Capture, Q Apply, Capture, Apply, and Replication Alert Monitor programs. You can provide this trace information to IBM Software Support for troubleshooting assistance. Run this command at an operating system prompt or in a shell script.

Syntax

►► *asntrc* —————►



On parameters:



Format parameters:



└─df─function_name|component_name diag_mask┘ └─holdlock┘

Change settings parameters:

└─d─diag_mask┘ └─df─function_name|component_name diag_mask┘

Parameters

Table 37 defines the invocation parameters for the `asnlrc` command.

Table 37. *asnlrc* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems

Parameter	Definition
<code>on</code>	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.
<code>-db db_name</code>	Specifies the name of the database to be traced: <ul style="list-style-type: none"> • 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.
<code>-qcap</code>	Specifies that a Q Capture program is to be traced. The Q Capture program is identified by the <code>-schema</code> parameter.
<code>-schema qcapture_schema</code>	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 <code>-qcap</code> parameter.
<code>-qapp</code>	Specifies that a Q Apply program is to be traced. The Q Apply program is identified by the <code>-schema</code> parameter.
<code>-schema qapply_schema</code>	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 <code>-qapp</code> parameter.
<code>-cap</code>	Specifies that a Capture program is to be traced. The Capture program is identified by the <code>-schema</code> parameter.
<code>-schema capture_schema</code>	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 <code>-cap</code> parameter.

Table 37. *asnlrc invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)*

Parameter	Definition
-app	Specifies that an Apply program is to be traced. The Apply program is identified by the -qualifier parameter.
-qualifier <i>apply_qualifier</i>	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 <i>monitor_qualifier</i>	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 <i>filename</i>	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.

Table 37. *asnlrc* invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)

Parameter	Definition
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.
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	<p>Specifies the types of trace records to be recorded by the trace facility. Trace records are categorized by a diagnostic mask number:</p> <ul style="list-style-type: none"> 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. <p>Important: The higher diagnostic mask numbers are <i>not</i> inclusive of the lower diagnostic mask numbers.</p> <p>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.</p> <p>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.</p>

Table 37. *asnlrc invocation parameter definitions for Linux, UNIX, Windows, and z/OS operating systems (continued)*

Parameter	Definition
<code>-df function_name component_name diag_mask</code>	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 asnlrc

The following examples illustrate how to use the `asnlrc` 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:

```
asnlrc 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:

```
asnlrc flw -db mydb -cap -schema myschema
```

To view the flow, basic, detailed, and performance data records only from the Capture log reader:

```
asnlrc fmt -db mydb -cap -schema myschema -d 0  
-df "Capture Log Read" 1,2,3,4
```

4. Stop the trace facility:

```
asnlrc 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:

```
asnlrc flw -fn myfile.trc
```

and

```
asnlrc 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:

```
asnlrc on -db mydb -mon -qualifier monq
```

Example 3

To trace only performance data of an Apply program:

```
asnlrc on -db mydb -app -qualifier aql -b 256k -fn myfile.trc -d 4
```

Example 4

To trace all flow and performance data of a Capture program:

```
asntrc on dbserve1 -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

```
▶▶ asntrep —DB=server—DB2_SUBSYSTEM=subsystem—SCHEMA=schema—
▶ —DIFF_SCHEMA=difference_table_schema—DIFF_TABLESPACE=tablespace—
▶ —WHERE=WHERE_clause—DIFF_PATH=log_path—PWDFILE=filename—
▶ —DIFF=table_name—
```

Parameters

Table 38 defines the invocation parameters for the `asntrep` command.

Table 38. *asntrep* invocation parameter definitions for Linux, UNIX, and Windows operating systems

Parameter	Definition
<code>DB=server</code>	<p>Specifies the DB2 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:</p> <p>Q replication The value is the name of the Q Capture server, which contains the <code>IBMQREP_SUBS</code> table.</p> <p>SQL replication The value is the name of the Apply control server, which contains the <code>IBMSNAP_SUBS_MEMBR</code> table.</p> <p>z/OS: The value of this parameter is a location name.</p>
<code>DB2_SUBSYSTEM=subsystem</code>	<p>On z/OS, specifies the name of the subsystem where the control tables are created.</p>
<code>SCHEMA=schema</code>	<p>Specifies the schema of the Q Capture control tables for Q replication, or the Apply control tables for SQL replication.</p>
<code>DIFF_SCHEMA=difference_table_schema</code>	<p>Specifies the schema that qualifies the difference table. The default is <code>ASN</code>.</p>
<code>DIFF_TABLESPACE=tablespace</code>	<p>Specifies the table space where a copy of the difference table is placed in the target database or subsystem. The copy is then used to repair the target table. If this parameter is not specified, the table will be created in the default table space in the database or subsystem in which the <code>asntrep</code> command was run.</p>

Table 38. *asntrep* invocation parameter definitions for Linux, UNIX, and Windows operating systems (continued)

Parameter	Definition
WHERE=WHERE_clause	<p>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:</p> <p>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.</p> <p>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.</p>
DIFF_PATH=log_path	<p>Specifies the location where you want the <i>asntrep</i> utility to write its log. The default value is the directory where you ran the command. The value must be an absolute path name. Use double quotation marks (") to preserve case.</p>
PWDFILE=filename	<p>Specifies the name of the password file that is used to connect to databases. If you do not specify a password file, the default value is <i>asnpwd.aut</i> (the name of the password file that is created by the <i>asnpwd</i> command). The <i>asntrep</i> utility searches for the password file in the directory that is specified by the DIFF_PATH parameter. If no value for the DIFF_PATH parameter is specified, the command searches for the password file in the directory where the command was run.</p>
DIFF=table_name	<p>Specifies the name of the table that was created in the source database using the <i>asntdiff</i> 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.</p>

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="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="set_name = 'my_set'  
and target_table = 'trg_table'" diff=sql_diff_table
```

Chapter 23. System commands for SQL replication (iSeries)

Some replication commands are specific to the iSeries operating system on iSeries servers. You can enter these commands at an operating system command prompt or through a command line program.

The following topics describe these commands.

ADDDPRREG: Adding a DPR registration (iSeries)

Use the Add DPR registration (ADDDPRREG) command to register a table as a source table for DB2 DataPropagator for iSeries.

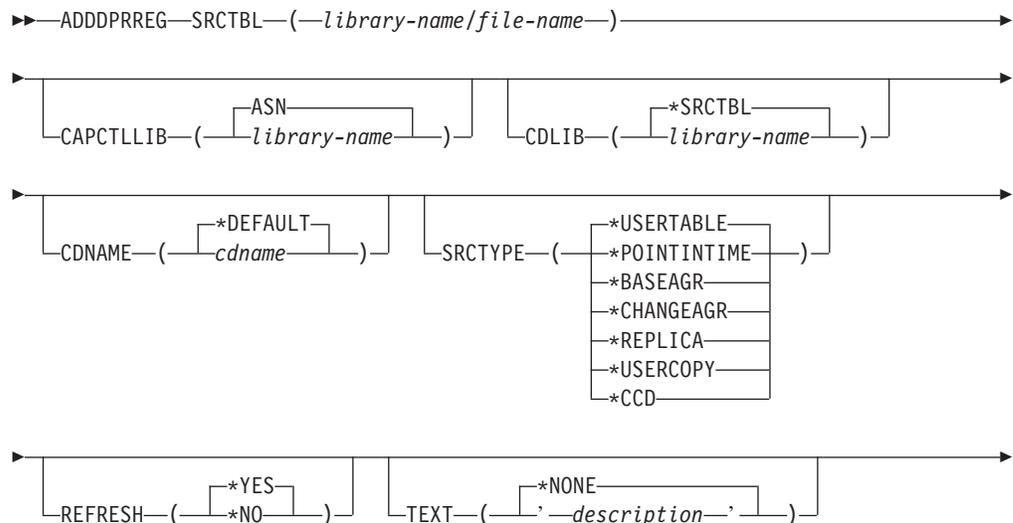
Use the Add DPR registration (ADDDPRREG) command to register a table as a source table for DB2 DataPropagator for iSeries.

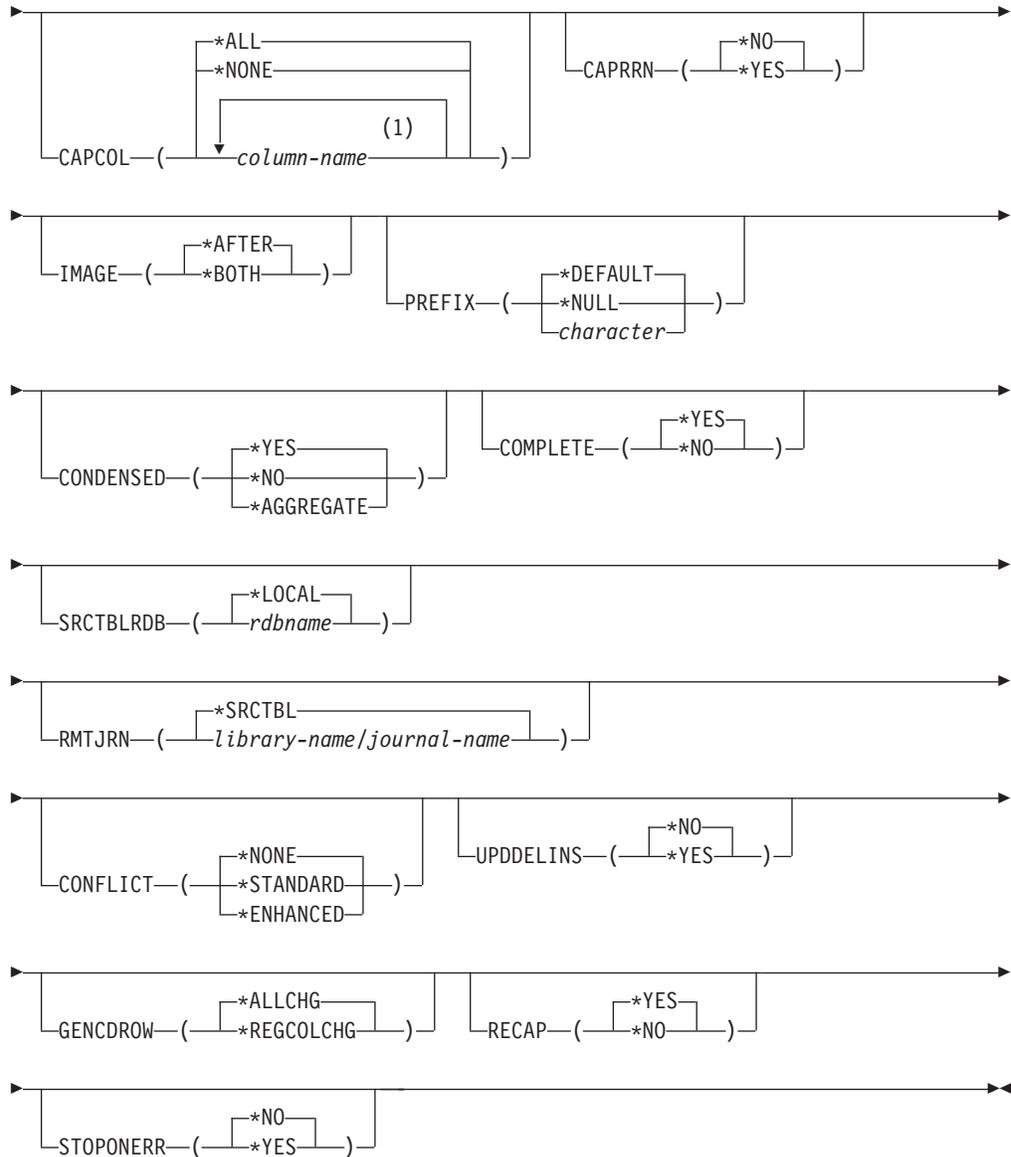
Restriction: You can register a table only if the ASN (Capture schema) library is in the same Auxiliary Pool (either base or independent ASP) where the ASN library is located.

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.

Syntax





Notes:

1 You can specify up to 300 column names.

Table 39 lists the invocation parameters.

Table 39. ADDDPREG command parameter definitions for iSeries

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 iSeries library or collection that is externally defined and in single format. This parameter is required. <i>library-name/file-name</i> Represents the qualified name of the table that you want to register.

Table 39. ADDDPREG command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
CAPCTLLIB	<p>Specifies the Capture schema, which is the name of the library in which the Capture control tables reside.</p> <p>ASN (default) The Capture control tables reside in the ASN library.</p> <p><i>library-name</i> The name of the library that contains the Capture control tables. You can create this library using the CRTDPRTBL command with the CAPCTLLIB parameter.</p>
CDLIB	<p>Specifies the library in which the change-data (CD) table for this registered source is created.</p> <p>*SRCTBL (default) Creates the CD table in the library in which the source table resides.</p> <p><i>library-name</i> Creates the CD table in this specified library name.</p>
CDNAME	<p>Specifies the name of the change-data (CD) table.</p> <p>*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.</p> <p><i>cdname</i> Creates the CD table with this specified name.</p>

Table 39. ADDDPREG command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
SRCTYPE	<p>Specifies the type of source table that you are registering. Choose a source type based on your replication configuration:</p> <ul style="list-style-type: none"> • 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. <p>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.</p> <p>*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.</p> <p>*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.</p> <p>*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.</p> <p>*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.</p> <p>*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.</p> <p>*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.</p> <p>*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:</p> <ul style="list-style-type: none"> • 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

Table 39. ADDDPREG command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
REFRESH	<p>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.</p> <p>*YES (default) Full refreshes are allowed.</p> <p>*NO Full refreshes are not allowed.</p> <p>If the target table is a base aggregate or change aggregate, you should set this parameter to *No.</p>
TEXT	<p>Specifies the textual description that is associated with this registration.</p> <p>*NONE (default) No description is associated with the entry.</p> <p><i>description</i> The textual description of this registration. You can enter a maximum of 50 characters and must enclose the text in single quotation marks.</p>
CAPCOL	<p>Specifies the columns for which changes are captured for this registered table.</p> <p>*ALL (default) Changes are captured for all columns.</p> <p>*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.</p> <p><i>column-name</i> The column names for which changes are captured. You can type up to 300 column names. Separate the column names with spaces.</p>
CAPRRN	<p>Specifies whether the relative record number (RRN) of each changed record is captured.</p> <p>*NO (default) The relative record number is not captured.</p> <p>*YES The relative record number is captured. An additional column called IBMQSQ_RRN is created in the change-data (CD) table.</p> <p>Set this parameter to *YES only if there are no unique keys in the source table.</p>

Table 39. ADDDPREG command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
IMAGE	<p>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.</p> <p>This IMAGE parameter is not valid when the CAPCOL parameter is set to *NONE.</p> <p>The source table must be journaled with *BOTH images even if you specify *AFTER on this parameter.</p> <p>*AFTER (default) The Capture program records only after images of the source table in the CD table.</p> <p>*BOTH The Capture program records both before images and after images of the source table in the CD table.</p>
PREFIX	<p>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.</p> <p>*DEFAULT (default) The default prefix (@) is used.</p> <p>*NULL No before images are captured. This value is not valid if the IMAGE parameter is set to *BOTH.</p> <p><i>character</i> Any single alphabetic character that is valid in an object name.</p>
CONDENSED	<p>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.</p> <p>*YES (default) The source table is condensed.</p> <p>*NO The source table is not condensed.</p> <p>*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</p>
COMPLETE	<p>Specifies whether the source table is complete, which means that the table contains a row for every primary key value of interest.</p> <p>*YES (default) The source table is complete.</p> <p>*NO The source table is not complete.</p>

Table 39. ADDDPREG command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
SRCTBLRDB	<p>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.</p> <p>*LOCAL (default) The source table resides locally (on the machine where you are running the ADDDPREG command).</p> <p><i>rdbname</i> 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.</p>
RMTJRN	<p>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.</p> <p>*SRCTBL (default) The remote journal name is the same as the journal name of the source table.</p> <p><i>library-name/journal-name</i> The qualified library and journal name that reside on this system and are used for journaling the remote source table.</p> <p>You can specify a remote journal name only if you specified a remote source table location using the SRCTBLRDB parameter.</p>
CONFLICT	<p>Specifies the conflict level that is used by the Apply program when detecting conflicts in a replica subscription.</p> <p>*NONE (default) No conflict detection.</p> <p>*STANDARD Moderate conflict detection. The Apply program searches for conflicts in rows that are already captured in the replica change-data (CD) tables.</p> <p>*ENHANCED Enhanced conflict detection. This option provides the best data integrity among all replicas and source tables.</p>
UPDDELINS	<p>Determines how the Capture program stores updated source data in the change-data (CD) table.</p> <p>*NO (default) The Capture program stores each source change in a single row in the CD table.</p> <p>*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.</p>

Table 39. ADDDPRREG command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
GENCDROW	<p>Specifies whether the Capture program captures changes from all rows in the source table.</p> <p>*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.</p> <p>*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.</p> <p>You cannot specify *REGCOLCHG if the CAPCOL parameter is set to *ALL or *NONE.</p>
RECAP	<p>Specifies whether the changes made by the Apply program are recaptured by the Capture program.</p> <p>*YES (default) Changes made to the source table by the Apply program are captured and entered into the change-data (CD) table.</p> <p>*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.</p>
STOPONERR	<p>Specifies whether the Capture program stops when it encounters an error.¹</p> <p>*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.</p> <p>*YES The Capture program issues messages and then stops when it encounters an error.</p>

Note:

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 200Axxx where xxx 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 iSeries documentation for the corrected action.
 - Use the Replication Center or the iSeries 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)
```

ADDDPRSUB: Adding a DPR subscription set (iSeries)

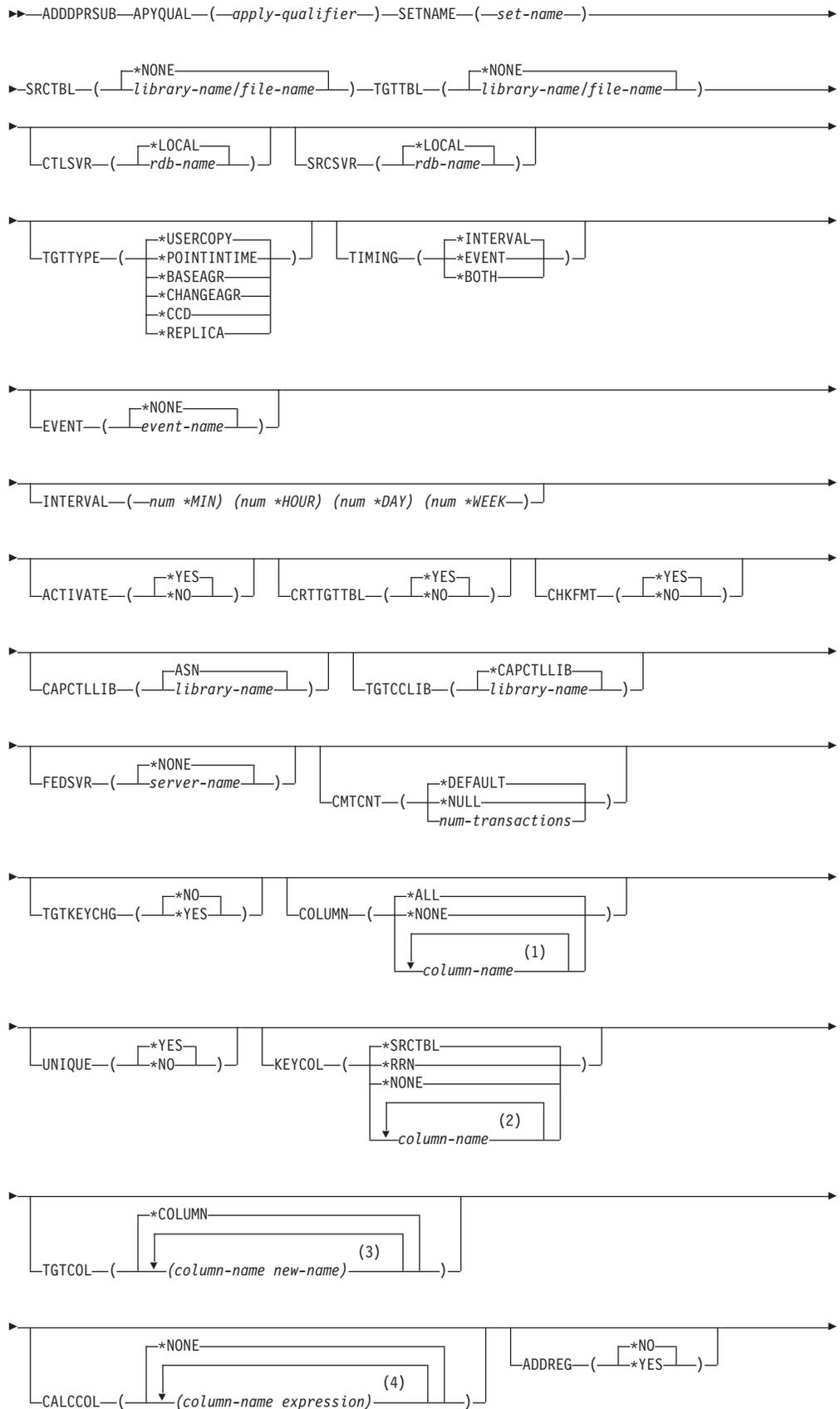
Use the Add DPR subscription set (ADDDPRSUB) command to create a subscription set with either one member or no members.

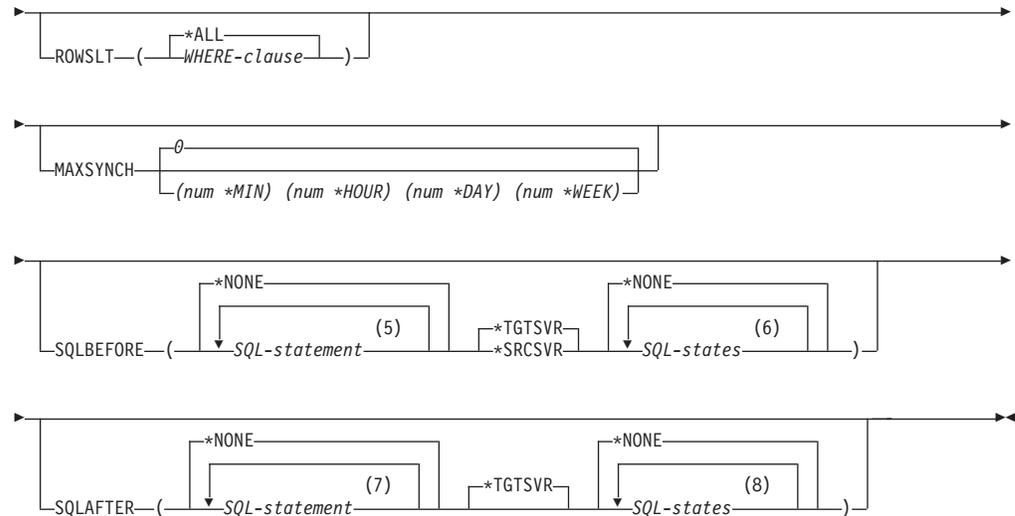
Use the Add DPR subscription set (ADDDPRSUB) command to create a subscription set with either one member or no members.

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.

Syntax





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 40 lists the invocation parameters.

Table 40. ADDDPRSUB command parameter definitions for iSeries

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. <i>apply-qualifier</i> The name of the Apply qualifier.
SETNAME	Specifies the subscription-set name. This parameter is required. <i>set-name</i> 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 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
SRCTBL	<p>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.</p> <p>*NONE (default) This subscription set does not have a source member. Use when creating a subscription set without members.</p> <p><i>library-name/file-name</i> The qualified name of the source table. Use when creating a subscription set with one member.</p>
TGTTBL	<p>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.</p> <p>*NONE (default) This subscription set does not have a target member. Use when creating a subscription set without members.</p> <p><i>library-name/file-name</i> The qualified name of the target table. Use when creating a subscription set with one member.</p>
CTLSVR	<p>Specifies the relational database name of the system that contains the Apply control tables.</p> <p>*LOCAL (default) The Apply control tables reside locally (on the machine from which you are running the ADDDPRSUB command).</p> <p><i>rdb-name</i> 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.</p>
SRCSVR	<p>Specifies the relational database name of the system that contains the Capture control tables.</p> <p>*LOCAL (default) The source table is registered on the local machine (the machine from which you are running the ADDDPRSUB command).</p> <p><i>rdb-name</i> 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.</p>

Table 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
TGTTYPE	<p>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.</p> <p>*USERCOPY (default)</p> <p>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.</p> <p>This value is not valid when a value of *RRN is specified on the KEYCOL parameter.</p> <p>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).</p> <p>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.</p> <p>*POINTINTIME</p> <p>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.</p> <p>*BASEAGR</p> <p>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.</p> <p>*CHANGEAGR</p> <p>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.</p> <p>*CCD</p> <p>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:</p> <ul style="list-style-type: none"> • IBMSNAP_INTENTSEQ • IBMSNAP_OPERATION • IBMSNAP_COMMITSEQ • IBMSNAP_LOGMARKER <p>*REPLICA</p> <p>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.</p>

Table 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
TIMING	<p>Specifies the type of timing (scheduling) that the Apply program uses to process the subscription set.</p> <p>*INTERVAL (default) The Apply program processes the subscription set at a specific time interval (for example, once a day).</p> <p>*EVENT The Apply program processes the subscription set when a specific event occurs.</p> <p>*BOTH The Apply program processes the subscription set either at a specific interval or when an event occurs, whichever occurs first.</p>
EVENT	<p>Specifies an event. The event that you enter must match an event name in the IBMSNAP_SUBS_EVENT) table.</p> <p>*NONE (default) No event is used.</p> <p><i>event-name</i> A unique character string that represents an event described in the IBMSNAP_SUBS_EVENT table.</p>
INTERVAL	<p>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:</p> <p>*MIN Minutes</p> <p>*HOUR Hours</p> <p>*DAY Days</p> <p>*WEEK Weeks</p> <p>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.</p>
ACTIVATE	<p>Specifies whether the subscription set is active. The Apply program does not process this subscription set unless this parameter is set to *YES.</p> <p>*YES (default) The subscription set is active.</p> <p>*NO The subscription set is not active.</p>

Table 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
CRTTGTTBL	<p>Specifies whether the target table (or view) is created.</p> <p>*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.</p> <p>*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.</p> <p>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.</p> <p>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.</p>
CHKFMT	<p>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.</p> <p>*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.</p> <p>*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.</p>
CAPCTLLIB	<p>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.</p> <p>ASN (default) The Capture control tables reside in the ASN library.</p> <p><i>library-name</i> The name of a library that contains the Capture control tables. This is the library in which the source table was registered.</p>

Table 40. ADDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
TGTCCLIB	<p>Specifies the target control library.</p> <p>*CAPCTLLIB (default) The target control library is the same library in which the Capture control tables reside.</p> <p><i>library-name</i> The name of a library that contains the target control tables.</p> <p>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.</p>
FEDSVR	<p>Specifies whether a federated database system is the source for this subscription set.</p> <p>*NONE (default) The source server is not a federated database system.</p> <p><i>server-name</i> The name of the federated database system for this subscription set (for non-DB2 relational sources).</p>
CMTCNT	<p>Specifies the commitment count, which is the number of transactions that the Apply program processes before a commit.</p> <p>*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.</p> <p>*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.</p> <p><i>num-transactions</i> 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.</p>

Table 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
TGTKEYCHG	<p>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:</p> <ul style="list-style-type: none"> • 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. <p>*NO (default) Updates to the source table are staged by the Capture program and processed by the Apply program to the target table.</p> <p>*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.</p>
COLUMN	<p>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.</p> <p>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.</p> <p>*ALL (default) All of the columns that you registered in the source are included in the target table.</p> <p>*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.</p> <p><i>column-name</i> The names of up to 300 source columns that you want to include in the target table. Separate the column names with spaces.</p>

Table 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
UNIQUE	<p>Specifies whether the target table has unique keys as indicated by the KEYCOL parameter.</p> <p>*YES (default)</p> <p>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.</p> <p>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.</p> <p>*NO</p> <p>The target table supports multiple changes per key. The changes are appended to the target table.</p> <p>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.</p>

Table 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
KEYCOL	<p>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.</p> <p>*SRCTBL (default)</p> <p>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:</p> <ul style="list-style-type: none"> • 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 <p>If you use a column as a key more than once and with different ordering, the target table key is defined with ascending order.</p> <p>*RRN</p> <p>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.</p> <p>*NONE</p> <p>The target copy does not contain a target key. You cannot specify *NONE if the target table type is *POINTINTIME, *REPLICA, or *USERCOPY.</p> <p><i>column-name</i></p> <p>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.</p>

Table 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
TGTCOL	<p>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.</p> <p>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.</p> <p>*COLUMN (default) The target columns are the same as the columns you specified in the COLUMN parameter.</p> <p><i>column-name</i> The column names from the source table that you want to change at the target. You can list up to 300 column names.</p> <p><i>new-name</i> 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.</p>
CALCCOL	<p>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.</p> <p>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.</p> <p>*NONE (default) No user-defined or calculated columns are included in the target table.</p> <p><i>column-name</i> The column names of the user-defined or calculated columns in the target table. You can list up to 100 column names.</p> <p><i>expression</i> The expressions for the user-defined or calculated columns in the target table. You can list up to 100 SQL column expressions.</p>
ADDREG	<p>Specifies whether the target table is automatically registered as a source table. Use this parameter to register CCD target type tables.</p> <p>*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.</p> <p>*YES The target table is registered as a source table. This command fails if you already registered the target table.</p> <p>Do not set this parameter to *YES if the target table type is *USERCOPY, *POINTINTIME, *BASEAGR, or *CHANGEAGR.</p> <p>If you set the CRTTGTTBL parameter to *NO, you must create the target table before attempting to register it as a source.</p>

Table 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
ROWSLT	<p>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.</p> <p>*ALL (default) The Apply program applies all changes in the CD table to the target table.</p> <p><i>WHERE-clause</i> 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.</p> <p>Note: The WHERE clause on this parameter is unrelated to any WHERE clauses specified on the SQLBEFORE or SQLAFTER parameters.</p>
MAXSYNCH	<p>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:</p> <p>*MIN Minutes</p> <p>*HOUR Hours</p> <p>*DAY Days</p> <p>*WEEK Weeks</p> <p>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.</p> <p>The default is zero (0), which indicates that all of the change data is to be applied.</p>

Table 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
SQLBEFORE	<p>Specifies the SQL statements that run before the Apply program refreshes the target table. This parameter has three elements:</p> <p>Element 1: SQL code</p> <p>*NONE (default) No SQL statement is specified.</p> <p><i>SQL-statement</i> 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.</p> <p>Element 2: Server to run on</p> <p>*TGTSVR (default) The SQL statement runs at the target server on which the target table is located.</p> <p>*SRCSVR The SQL statement runs at the Capture control server on which the source table is located.</p> <p>Element 3: Allowed SQLSTATE values</p> <p>*NONE (default) Only an SQLSTATE value of 00000 is considered successful.</p> <p><i>SQL-states</i> 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.</p> <p>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.</p>

Table 40. ADDDPRSUB command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
SQLAFTER	<p>Specifies SQL statements that run after the Apply program refreshes the target table. This parameter has three elements:</p> <p>Element 1: SQL code</p> <p>*NONE (default) No SQL statement is specified.</p> <p><i>SQL-statement</i> 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.</p> <p>Element 2: Server to run on</p> <p>*TGTSVR (default) The SQL statement runs at the target server on which the target table is located.</p> <p>Element 3: Allowed SQLSTATE values</p> <p>*NONE (default) Only an SQLSTATE value of 00000 is considered successful.</p> <p><i>SQL-states</i> 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.</p> <p>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.</p>

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)
  TGTBL(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) TGTBL(*NONE)
```

ADDDPRSUBM: Adding a DPR subscription-set member (iSeries)

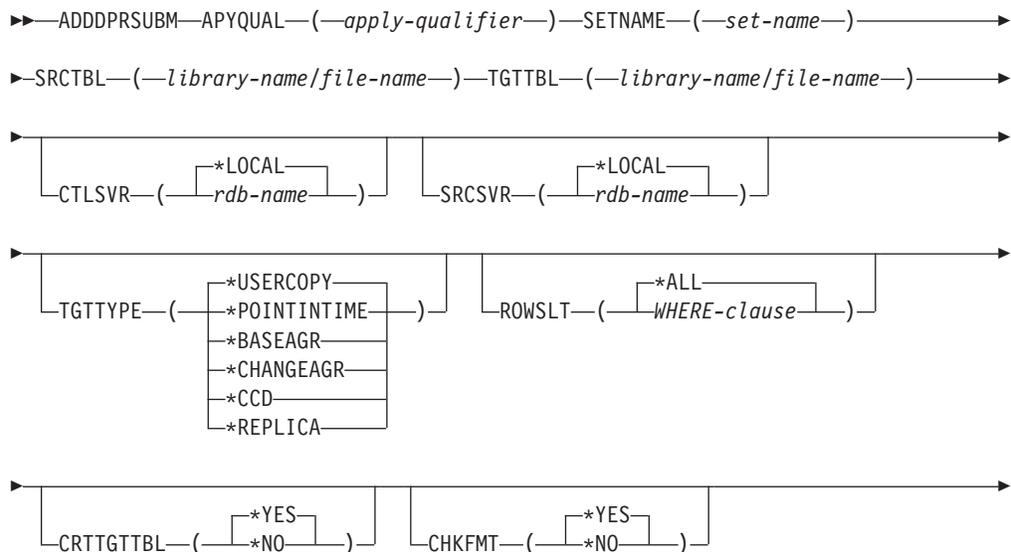
Use the Add DPR subscription-set member (ADDDPRSUBM) command to add a member to an existing subscription set.

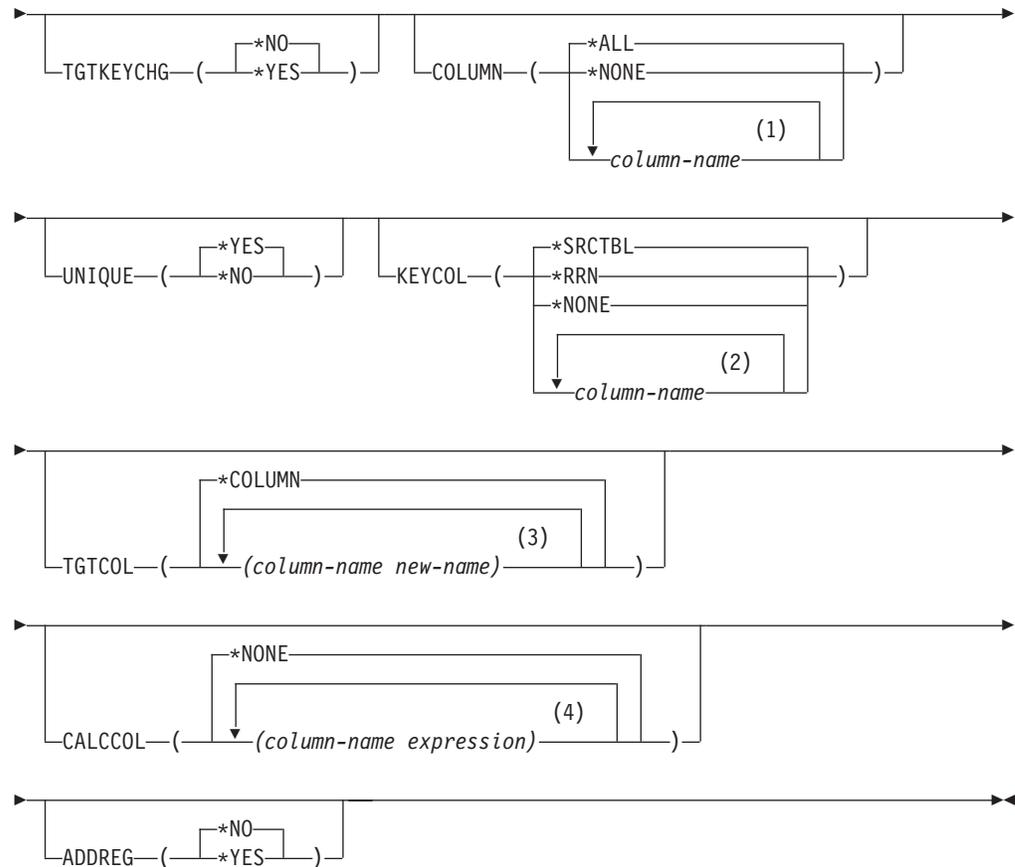
Use the Add DPR subscription-set member (ADDDPRSUBM) command to add a member to an existing subscription set. You can create the subscription set with the ADDDPRSUB command, with the system commands on UNIX, Windows, or z/OS, or through the Replication Center. All the source tables in the subscription set must already be journaled and 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.

Syntax





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 41 lists the invocation parameters.

Table 41. ADDDPRSUBM command parameter definitions for iSeries

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. <i>apply-qualifier</i> The name of the Apply qualifier.
SETNAME	Specifies the name of the subscription set. This parameter is required. <i>set-name</i> 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.

Table 41. ADDDPRSUBM command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
SRCTBL	<p>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.</p> <p><i>library-name/file-name</i> The qualified name of the source table.</p>
TGTTBL	<p>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.</p> <p><i>library-name/file-name</i> The qualified name of the target table.</p>
CTLSVR	<p>Specifies the relational database name of the system that contains the Apply control tables.</p> <p>*LOCAL (default) The Apply control tables reside locally (on the machine from which you are running the ADDDPRSUBM command).</p> <p><i>rdb-name</i> 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.</p>
SRCSVR	<p>Specifies the relational database name of the system that contains the Capture control tables.</p> <p>*LOCAL (default) The source table is registered on the local machine (the machine from which you are running the ADDDPRSUBM command).</p> <p><i>rdb-name</i> 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.</p>

Table 41. ADDDPRSUBM command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
TGTTYPE	<p>Specifies the target table type. These are SQL 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.</p> <p>*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.</p> <p>This value is not valid when a value of *RRN is specified on the KEYCOL parameter.</p> <p>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).</p> <p>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.</p> <p>*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.</p> <p>*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.</p> <p>*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.</p> <p>*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:</p> <ul style="list-style-type: none"> • IBMSNAP_INTENTSEQ • IBMSNAP_OPERATION • IBMSNAP_COMMITSEQ • IBMSNAP_LOGMARKER <p>*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.</p>

Table 41. ADDDPRSUBM command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
ROWSLT	<p>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.</p> <p>*ALL (default) The Apply program applies all changes in the CD table to the target table.</p> <p><i>WHERE-clause</i> 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.</p> <p>Note: The WHERE clause on this parameter is unrelated to any WHERE clauses specified on the SQLBEFORE or SQLAFTER parameters.</p>
CRTTGTTBL	<p>Specifies whether the target table (or view) is created.</p> <p>*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.</p> <p>*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.</p> <p>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.</p> <p>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.</p>
CHKFMT	<p>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.</p> <p>*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.</p> <p>*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.</p>

Table 41. *ADDDPRSUBM* command parameter definitions for *iSeries* (continued)

Parameter	Definition and prompts
TGTKEYCHG	<p>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:</p> <ul style="list-style-type: none"> • 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. <p>*NO (default) Updates to the source table are staged by the Capture program and processed by the Apply program to the target table.</p> <p>*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.</p>
COLUMN	<p>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.</p> <p>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.</p> <p>*ALL (default) All of the columns that you registered in the source are included in the target table.</p> <p>*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.</p> <p><i>column-name</i> The names of up to 300 source columns that you want to include in the target table. Separate the column names with spaces.</p>

Table 41. ADDDPRSUBM command parameter definitions for iSeries (continued)

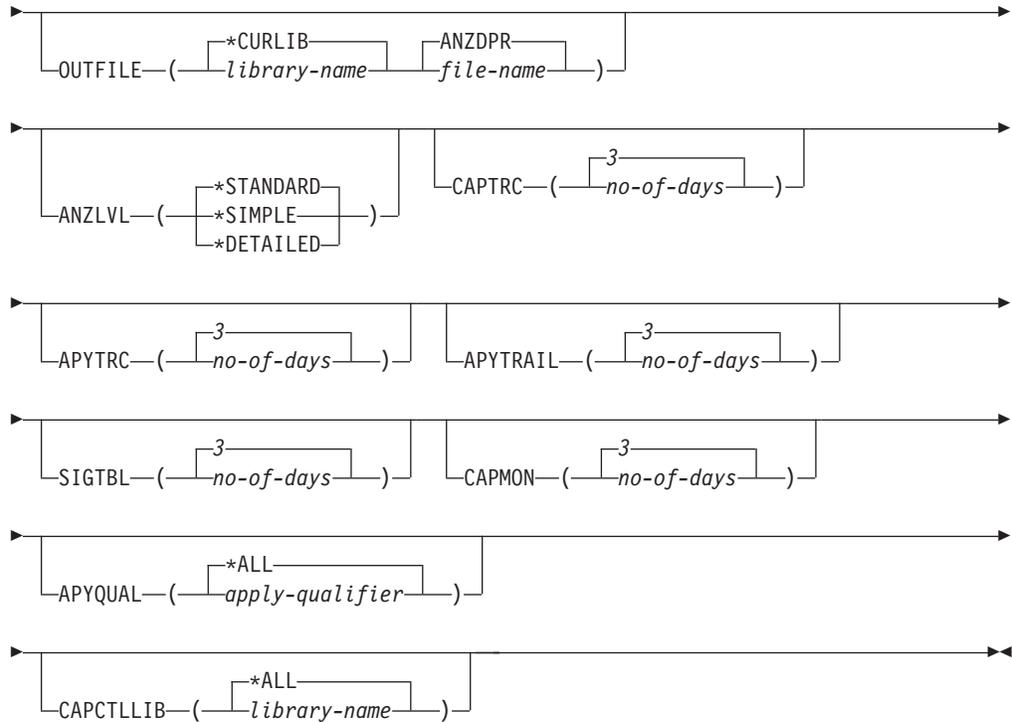
Parameter	Definition and prompts
UNIQUE	<p>Specifies whether the target table has unique keys as indicated by the KEYCOL parameter.</p> <p>*YES (default)</p> <p>The target table supports one net change per key; only one row exists in the target table for that key regardless of how many changes are made to the key.</p> <p>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.</p> <p>*NO</p> <p>The target table supports multiple changes per key. The changes are appended to the target table.</p> <p>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.</p>

Table 41. ADDDPRSUBM command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
KEYCOL	<p>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.</p> <p>*SRCTBL (default)</p> <p>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:</p> <ul style="list-style-type: none"> • 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 <p>If you use a column as a key more than once and with different ordering, the target table key is defined with ascending order.</p> <p>*RRN</p> <p>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.</p> <p>*NONE</p> <p>The target copy does not contain a target key. You cannot specify *NONE if the target table type is *POINTINTIME, *REPLICA, or *USERCOPY.</p> <p><i>column-name</i></p> <p>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.</p>

Table 41. ADDDPRSUBM command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
TGTCOL	<p>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.</p> <p>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.</p> <p>*COLUMN (default) The target columns are the same as the columns you specified in the COLUMN parameter.</p> <p><i>column-name</i> The column names from the source table that you want to change at the target. You can list up to 300 column names.</p> <p><i>new-name</i> 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.</p>
CALCCOL	<p>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.</p> <p>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.</p> <p>*NONE (default) No user-defined or calculated columns are included in the target table.</p> <p><i>column-name</i> The column names of the user-defined or calculated columns in the target table. You can list up to 100 column names.</p> <p><i>expression</i> The expressions for the user-defined or calculated columns in the target table. You can list up to 100 SQL column expressions.</p>
ADDREG	<p>Specifies whether the target table is automatically registered as a source table. Use this parameter to register CCD target type tables.</p> <p>*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.</p> <p>*YES The target table is registered as a source table. This command fails if you already registered the target table.</p> <p>Do not set this parameter to *YES if the target table type is *USERCOPY, *POINTINTIME, *BASEAGR, or *CHANGEAGR.</p> <p>If you set the CRTTGTTBL parameter to *NO, you must create the target table before attempting to register it as a source.</p>



Notes:

1 You can specify up to 10 databases.

Table 42 lists the invocation parameters.

Table 42. ANZDPR command parameter definitions for iSeries

Parameter	Definition and prompts
RDB	<p>Specifies the databases to be analyzed.</p> <p>*LOCAL (default) The database on your local system.</p> <p><i>rdb-name</i> The RDB Directory Entry name, which indicates the database.</p> <p>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.</p>

Table 42. ANZDPR command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
OUTFILE	<p>Specifies the library and file name used to store the analyzer output. This command writes the output to an HTML file.</p> <p>*CURLIB (default) The current library.</p> <p><i>library-name</i> The name of the library.</p> <p>ANZDPR (default) The output is written to an HTML file named ANZDPR.</p> <p><i>file-name</i> The name of the HTML output file.</p> <p>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).</p>
ANZLVL	<p>Specifies the level of analysis to be reported. The level of analysis can be:</p> <p>*STANDARD (default) Generates a report that includes the contents of the control tables as well as Capture and Apply program status information.</p> <p>*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.</p> <p>*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.</p>
CAPTRC	<p>Specifies the date range (0 to 30 days) of entries to be reported from the IBMSNAP_CAPTRACE table. The default is 3.</p> <p><i>no-of-days</i> The number of days to be reported.</p>
APYTRC	<p>Specifies the date range (0 to 30 days) of entries to be reported from the IBMSNAP_APPLYTRACE table. The default is 3.</p> <p><i>no-of-days</i> The number of days to be reported.</p>
APYTRAIL	<p>Specifies the date range (0 to 30 days) of entries to be reported from the IBMSNAP_APPLYTRAIL table. The default is 3.</p> <p><i>no-of-days</i> The number of days to be reported.</p>
SIGTBL	<p>Specifies the date range (0 to 30 days) of entries to be reported from the IBMSNAP_SIGNAL table. The default is 3.</p> <p><i>no-of-days</i> The number of days to be reported.</p>
CAPMON	<p>Specifies the date range (0 to 30 days) of entries to be reported from the IBMSNAP_CAPMON table. The default is 3.</p> <p><i>no-of-days</i> The number of days to be reported.</p>

Table 42. ANZDPR command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
APYQUAL	<p>Specifies the Apply qualifiers to be analyzed.</p> <p>*ALL (default) All Apply qualifiers are analyzed.</p> <p><i>apply-qualifier</i> The name of the Apply qualifier to be analyzed. You can enter up to 10 Apply qualifiers.</p>
CAPCTLLIB	<p>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.</p> <p>*ALL (default) All of the Capture control libraries will be analyzed.</p> <p><i>library-name</i> The name of the specific Capture control library that you want to analyze.</p>

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
```

CHGDPRCAPA: Changing DPR Capture attributes (iSeries)

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 IBMSNAP_CAPPARMS table.

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 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.

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.

Syntax

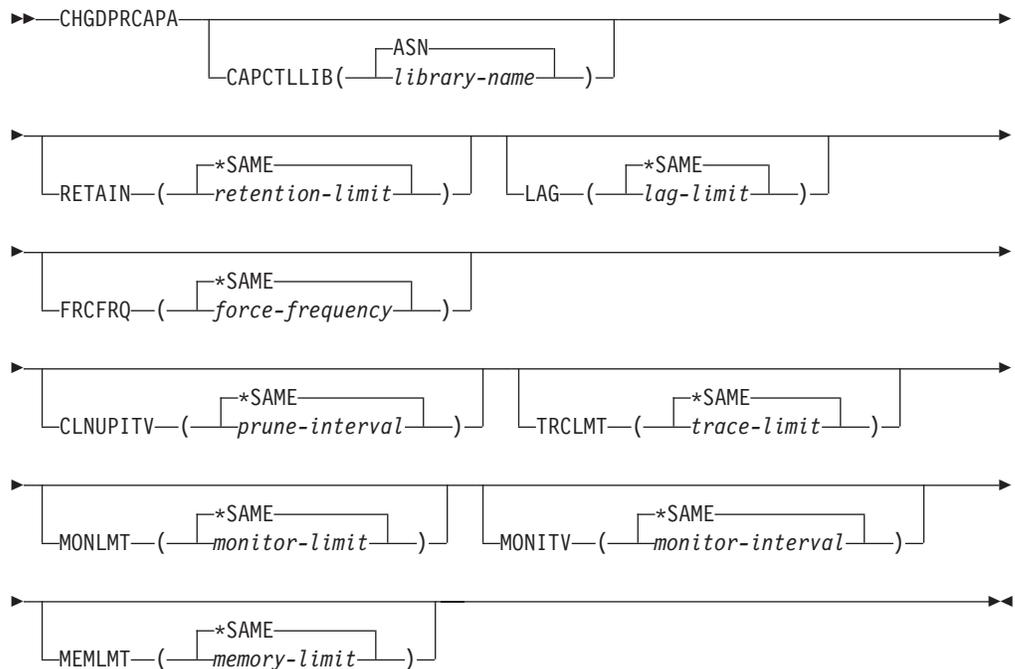


Table 43 lists the invocation parameters.

Table 43. CHGDPRCAPA command parameter definitions for iSeries

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. <i>library-name</i> The name of a library that contains the Capture control tables.

Table 43. CHGDPRCAPA command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
RETAIN	<p>Specifies the new retention limit, which is the number of minutes that data is retained in the change-data (CD), unit-of-work (UOW), IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN tables before this data is removed. This value is stored in the RETENTION_LIMIT column of the IBMSNAP_CAPPARMS table.</p> <p>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.</p> <p>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.</p> <p>The default is 10 080 minutes (seven days). The maximum is 35000000 minutes.</p> <p>*SAME (default) This value is not changed.</p> <p><i>retention-limit</i> The new retention limit value.</p>
LAG	<p>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 IBMSNAP_CAPPARMS table.</p> <p>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.</p> <p>The default is 10 080 minutes (seven days). The maximum is 35000000 minutes.</p> <p>*SAME (default) This value is not changed.</p> <p><i>lag-limit</i> The new lag limit value.</p>

Table 43. CHGDPRCAPA command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
FRCFRQ	<p>Specifies how often (from 30 to 600 seconds) the Capture program writes changes to the change-data (CD) and UOW tables. This value is stored in the COMMIT_INTERVAL column of the IBMSNAP_CAPPARMS table.</p> <p>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.</p> <p>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.</p> <p>The default is 30 seconds.</p> <p>*SAME (default) This value is not changed.</p> <p><i>force-frequency</i> 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.</p>
CLNUPITV	<p>Specifies the maximum amount of time (in hours) before the Capture program prunes old records from the change-data (CD), UOW, IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables.</p> <p>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.)</p> <p>The value of this parameter is automatically converted from hours to seconds and is stored in the PRUNE_INTERVAL column of the 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.</p> <p>*SAME (default) This Capture attribute value is not changed.</p> <p><i>prune-interval</i> The pruning interval expressed as a specific number of hours (1 to 100).</p>

Table 43. CHGDPRCAPA command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
TRCLMT	<p>Specifies the trace limit (in minutes). This value is stored in the TRACE_LIMIT column of the IBMSNAP_CAPPARMS table.</p> <p>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).</p> <p>*SAME (default) This value is not changed.</p> <p><i>trace-limit</i> The number of minutes of trace data kept in the IBMSNAP_CAPTRACE table after pruning.</p>
MONLMT	<p>Specifies the monitor limit (in minutes). This value is stored in the MONITOR_LIMIT column of the IBMSNAP_CAPPARMS table.</p> <p>The Capture program prunes any IBMSNAP_CAPMON rows that are older than the monitor limit.</p> <p>The default is 10 080 minutes (seven days of monitor entries).</p> <p>*SAME (default) This value is not changed.</p> <p><i>monitor-limit</i> The number of minutes of monitor data kept in the IBMSNAP_CAPMON table after pruning.</p>
MONITV	<p>Specifies how frequently (in seconds) the Capture program inserts rows into the IBMSNAP_CAPMON table. This value is stored in the MONITOR_INTERVAL column of the IBMSNAP_CAPPARMS table.</p> <p>The default is 300 seconds (five minutes).</p> <p>*SAME (default) This value is not changed.</p> <p><i>monitor-interval</i> 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.</p>
MEMLMT	<p>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 IBMSNAP_CAPPARMS table.</p> <p>The default is 32 megabytes.</p> <p>*SAME (default) This value is not changed.</p> <p><i>memory-limit</i> The maximum number of megabytes for memory.</p>

Examples for CHGDPRCAPA

The following examples illustrate how to use the CHGDPRCAPA command.

Example 1:

Table 44 lists the invocation parameters.

Table 44. CRTDPRTBL command parameter definitions for iSeries

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. <i>library-name</i> 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)
```

ENDDPRAPY: Stopping Apply (iSeries)

Use the End DPR Apply (ENDDPRAPY) command to stop an Apply program on your local system.

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.

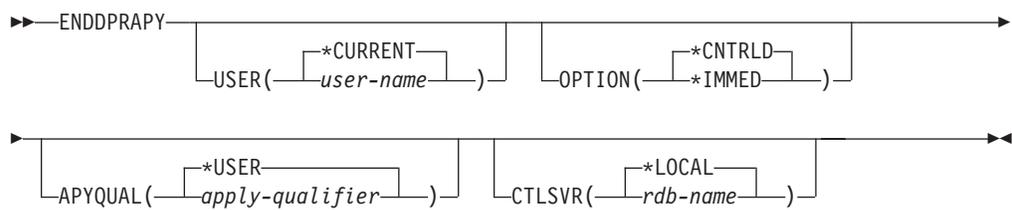


Table 45 lists the invocation parameters.

Table 45. ENDDPRAPY command parameter definitions for iSeries

Parameter	Definition and prompts
USER	<p>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.</p> <p>*CURRENT (default) The Apply program of the user associated with the current job.</p> <p><i>user-name</i> The Apply program of the specified user.</p> <p>When prompting on the ENDDPRAPY command, you can press the F4 key to see a list of users who defined subscriptions.</p>
OPTION	<p>Specifies how to stop the Apply program.</p> <p>*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.</p> <p>*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.)</p> <p>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.</p> <p>You might find that a subscription set is considered IN USE (the STATUS column in the 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.</p>
APYQUAL	<p>Specifies the Apply qualifier that is used by the Apply program.</p> <p>*USER (default) The user name specified on the USER parameter is the Apply qualifier.</p> <p><i>apply-qualifier</i> 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 IBMSNAP_SUBS_SET table with this value in the APPLY_QUAL column.</p> <p>When prompting on the ENDDPRAPY command, you can press the F4 key to see a list of Apply qualifier names with existing subscriptions.</p>

Table 45. ENDDPRAPY command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
CTLSVR	<p>Specifies the relational database name of the system that contains the Apply control tables.</p> <p>*LOCAL (default) The Apply control tables reside locally (from the machine on which you are running the ENDDPRAPY command).</p> <p><i>rdb-name</i> 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.</p> <p>When prompting on the ENDDPRAPY command, you can press the F4 key to choose from the list of databases in the RDB directory.</p>

Usage notes

The ENDDPRAPY command uses the value of the **APYQUAL** and **CTLSVR** parameters to search the 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)
```

ENDDPRCAP: Stopping Capture (iSeries)

Use the End DPR Capture (ENDDPRCAP) command to stop the Capture program.

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.

Syntax

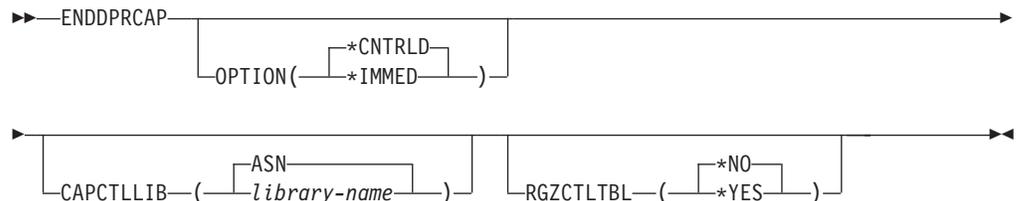


Table 46 lists the invocation parameters.

Table 46. ENDDPRCAP command parameter definitions for iSeries

Parameter	Definition and prompts
OPTION	Specifies how to stop the Capture program. <ul style="list-style-type: none"> *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 IBMSNAP_REGISTER table, which stores the registration information of the source tables. <ul style="list-style-type: none"> ASN (default) The Capture control tables are in the ASN library. The ASN library is the default library. <i>library-name</i> The name of a library that contains the Capture control tables.

Table 46. ENDDPRCAP command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
RGZCTLTLBL	<p>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.</p> <p>*NO (default) The RGZPFM command is not performed.</p> <p>*YES The RGZPFM command is performed.</p>

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 QDP4nnnnnn, 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) RGZCTLTLBL(*NO)
```

Example 2:

To end the Capture program immediately for the Capture schema BSN:

```
ENDDPRCAP OPTION(*IMMED) CAPCTLLIB(BSN) RGZCTLTLBL(*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) RGZCTLTLBL(*YES)
```

GRTDPRAUT: Authorizing users (iSeries)

Use the Grant DPR Authority (GRTDPRAUT) command to authorize a list of users to access the replication control tables in order to run the Capture and Apply programs.

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.

Syntax

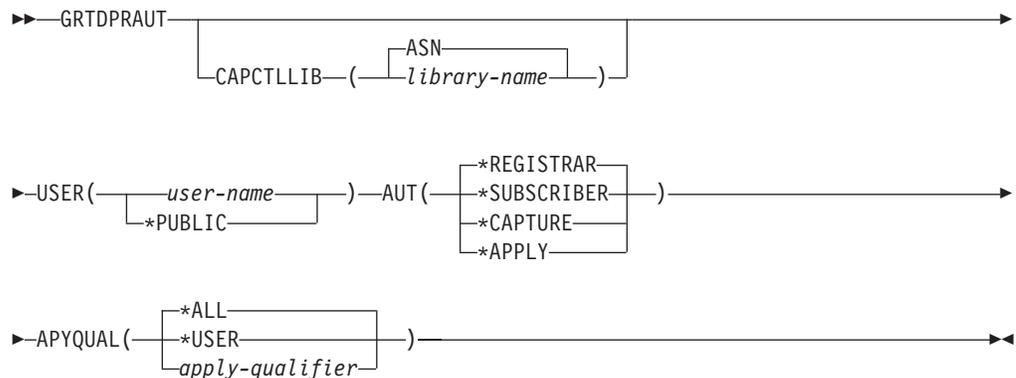


Table 47 lists the invocation parameters.

Table 47. GRTDPRAUT command parameter definitions for iSeries

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. <i>library-name</i> The name of the library that contains the replication control tables.

Table 47. GRTDPRAUT command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
USER	<p>Specifies the users who have authority.</p> <p><i>user-name</i> The names of up to 50 users who have authority.</p> <p>*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.</p>
AUT	<p>Specifies the type of authority being granted.</p> <p>*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 48 on page 335.</p> <p>*SUBSCRIBER The users are granted authority to define, change, and remove subscription sets. For a complete list of authorities with AUT(*SUBSCRIBER), see Table 49 on page 336.</p> <p>*CAPTURE The users are granted authority to run the Capture program. For a complete list of authorities granted with AUT(*CAPTURE), see Table 50 on page 337.</p> <p>*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 51 on page 339.</p>

Table 47. GRTDPRAUT command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
APYQUAL	<p>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.</p> <p>*ALL (default) The user is granted authority to run the Apply program or to define and remove subscription sets for all Apply qualifiers.</p> <p>*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.</p> <p><i>apply-qualifier</i> 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.</p> <ul style="list-style-type: none"> • The user is granted authority to all replication sources, change-data (CD) tables, and consistent-change data (CCD) tables associated with records in the 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 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 48. Authorities granted with GRTDPRAUT AUT(*REGISTRAR)

Library	Object	Type	Authorizations
QSYS	capctllib	*LIB	*USE, *ADD
capctllib ¹	QSQRJRN	*JRN	*OBJOPR, *OBJMGT
capctllib ¹	QZS8CTLBLK	*USRSPC	*CHANGE
capctllib ¹	IBMSNAP_REGISTER	*FILE	*OBJOPR, *READ, *ADD, *UPDT, *DLT

Table 48. 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

Note:

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 49. 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 49. 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

Note:

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 50. 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 50. 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

Note:

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 51. 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 51. Authorities granted with GRTPRAUT 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

Note:

1. The entry *capctllib* in the Library column refers to the value passed to the **CAPCTLLIB** parameter of the GRTPRAUT command; this command updates authority to only one Capture control library at a time.

Examples for GRTPRAUT

The following examples illustrate how to use the GRTPRAUT command.

Example 1:

To authorize a user named USER1 to define and modify registrations:

```
GRTPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*REGISTRAR)
```

Example 2:

To authorize a user named USER1 to define and modify subscription sets:

```
GRTPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*SUBSCRIBER)
```

Example 3:

To authorize a user named USER1 to run Capture programs:

```
GRTPRAUT 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:

```
GRTPRAUT 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:
GRTPRAUT CAPCTLLIB(ASN) USER(USER1) AUT(*APPLY) APYQUAL(A1)
2. Run the appropriate GRTPRAUT 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)

INZDPRCAP: Reinitializing DPR Capture (iSeries)

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.

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.

Syntax

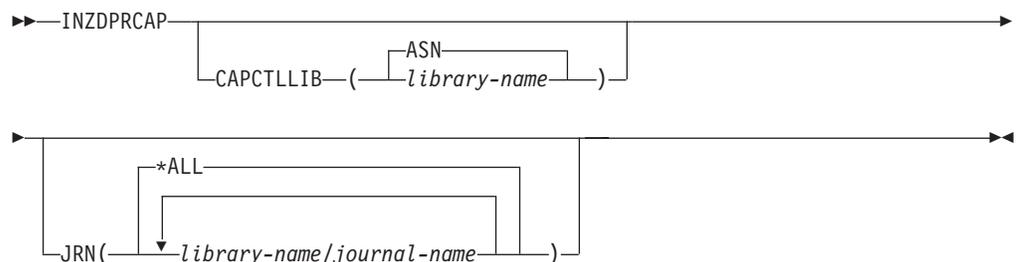


Table 52 on page 342 lists the invocation parameters.

Table 52. INZDPRCAP command parameter definitions for iSeries

Parameter	Definition and prompts
CAPCTLLIB	<p>Specifies the Capture schema, which is the name of the library in which the Capture control tables reside.</p> <p>ASN (default) The Capture control tables reside in the ASN library. The ASN library is the default library.</p> <p><i>library-name</i> The name of a library that contains the Capture control tables.</p>
JRN	<p>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.</p> <p>*ALL (default) The Capture program works with all the journals.</p> <p><i>library-name/journal-name</i> The qualified name of the journal that you want the Capture program to work with.</p>

Examples for INZDPRCAP

The following examples illustrate how to use the INZDPRCAP command.

Example 1:

To initialize a Capture program using the QSQRJRN journal under a library named TRAINING:

```
INZDPRCAP CAPCTLLIB(ASN) JRN(TRAINING/QSQRJRN)
```

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.

OVRDPRCAPA: Overriding DPR Capture attributes (iSeries)

Use the Override DPR Capture attributes (OVRDPRCAPA) command to alter the behavior of a running Capture program.

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 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.

Syntax

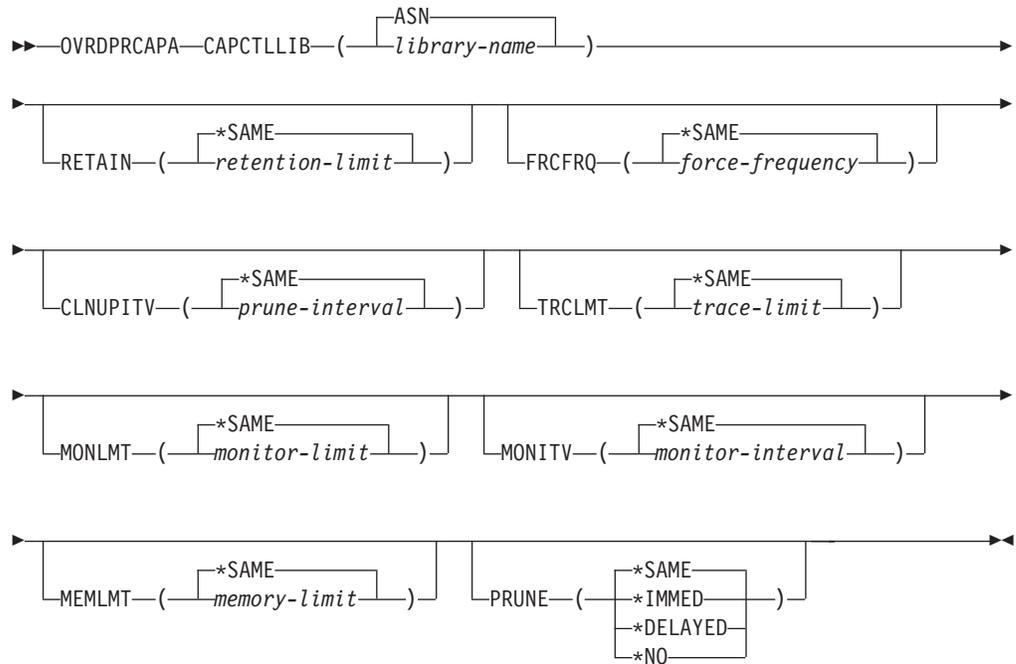


Table 53 lists the invocation parameters.

Table 53. OVRDPRCAPA command parameter definitions for iSeries

Parameter	Definition and prompts
CAPCTLLIB	<p>Specifies the Capture schema, which is the name of the library in which the Capture control tables reside. This library includes the IBMSNAP_REGISTER table, which stores the registration information of the source tables. This parameter is required.</p> <p>ASN (default) The Capture control tables reside in the ASN library.</p> <p><i>library-name</i> The name of a library that contains the Capture control tables. You can create this library using the CRTDPRTBL command with the CAPCTLLIB parameter.</p>

Table 53. OVRDPRCAPA command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
RETAIN	<p>Specifies the number of minutes that data is retained in the change-data (CD), UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHTKN tables before the data is removed.</p> <p>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.</p> <p>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.</p> <p>The default is 10 080 minutes (seven days). The maximum is 35000000 minutes.</p> <p>*SAME (default) This value is not changed.</p> <p><i>retention-limit</i> The new retention limit value.</p>
FRCFRQ	<p>Specifies how often (from 30 to 600 seconds) the Capture program writes changes to the change-data (CD) and unit-of-work (UOW) tables.</p> <p>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.</p> <p>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.</p> <p>The default is 30 seconds.</p> <p>*SAME (default) This value is not changed.</p> <p><i>force-frequency</i> 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.</p>

Table 53. OVRDPRCAPA command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
CLNUPITV	<p>Specifies the maximum amount of time (in hours) before the Capture program prunes old records from the change-data (CD), unit-of-work (UOW), IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables.</p> <p>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.</p> <p>(Use the STRDPRCAP command to set the RETAIN, MONLMT, and TRCLMT parameters for a Capture program.)</p> <p>The value of the CLNUPITV parameter is automatically converted from hours to seconds and is stored in the PRUNE_INTERVAL column of the IBMSNAP_CAPPARMS table.</p> <p>*SAME (default) This Capture attribute value is not changed.</p> <p><i>prune-interval</i> The pruning interval expressed as a specific number of hours (1 to 100).</p>
TRCLMT	<p>Specifies the trace limit, which indicates how frequently the IBMSNAP_CAPTRACE table is pruned.</p> <p>*SAME (default) The Capture program continues using the current trace limit value.</p> <p><i>trace-limit</i> The number of minutes between each pruning operation of the IBMSNAP_CAPTRACE table.</p>
MONLMT	<p>Specifies the monitor limit, which indicates how frequently the IBMSNAP_CAPMON table is pruned.</p> <p>*SAME (default) The Capture program continues using the current monitor limit value.</p> <p><i>monitor-limit</i> The number of minutes between each pruning operation of the IBMSNAP_CAPMON table.</p>
MONITV	<p>Specifies the monitor interval (in seconds), which indicates how frequently the Capture program inserts rows into the IBMSNAP_CAPMON table.</p> <p>*SAME (default) The Capture program continues using the current monitor interval value.</p> <p><i>monitor-interval</i> 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.</p>

Table 53. OVRDPRCAPA command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
MEMLMT	<p>Specifies the maximum size (in megabytes) of memory that the Capture journal job can use.</p> <p>*SAME (default) The Capture program continues using the current memory limit value.</p> <p><i>memory-limit</i> The maximum number of megabytes for memory.</p>
PRUNE	<p>Use this parameter to change the way that the Capture program prunes rows from the change-data (CD), unit-of-work (UOW), IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables.</p> <p>*SAME (default) The Capture program continues using the pruning parameters that you specified when you started the STRDPRCAP command.</p> <p>*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.</p> <p>*DELAYED The Capture program prunes the old rows at the end of the specified pruning interval.</p> <p>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) <i>does</i> initiate pruning if you set the second part of the CLNUPITV parameter to *NO when you started the STRDPRCAP command.</p> <p>*NO The Capture program does not initiate pruning. This value overrides the CLNUPITV parameter setting from the STRDPRCAP command.</p>

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)
```

RMVDPRREG: Removing a DPR registration (iSeries)

Use the Remove DPR registration (RMVDPRREG) command to remove a single source table from the IBMSNAP_REGISTER table so that the source table is no longer used for replication.

Use the Remove DPR registration (RMVDPRREG) command to remove a single source table from the 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.

Syntax

```
►► RMVDPRREG SRCTBL (—library-name/file-name—)
└──┬──────────────────────────────────────────────────────────────────────────────────┘
└──┬──────────────────────────────────────────────────────────────────────────────────┘
    └──┬──────────────────────────────────────────────────────────────────────────────────┘
        CAPCTLLIB (—ASN—)
                   └──┬──────────────────────────────────────────────────────────────────┘
                       library-name
```

Table 54 lists the invocation parameters.

Table 54. RMVDPRREG command parameter definitions for iSeries

Parameter	Definition and prompts
SRCTBL	Identifies the registration that you want to remove. This is a required parameter. <i>library-name/file-name</i> 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. <i>library-name</i> 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)
```

RMVDPRSUB: Removing a DPR subscription set (iSeries)

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.

Syntax

```

▶▶ RMVDPRSUB APYQUAL (—apply-qualifier—) SETNAME (—set-name—)
|
| CTLSVR (—rdb-name—) | RMVREG (—*NO—) |
| *LOCAL | *YES |
|
| DLTGTBL (—*NO—) | RMVMBRS (—*NO—) |
| *YES | *YES |

```

Table 55 lists the invocation parameters.

Table 55. RMVDPRSUB command parameter definitions for iSeries

Parameter	Definition and prompts
APYQUAL	Specifies the Apply qualifier that the Apply program uses to identify the subscription set. This parameter is required. <i>apply-qualifier</i> The name of the Apply qualifier.
SETNAME	Specifies the name of the subscription set. This parameter is required. <i>set-name</i> 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.
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). <i>rdb-name</i> 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.

Table 55. *RMVDPRSUB* command parameter definitions for *iSeries* (continued)

Parameter	Definition and prompts
RMVREG	<p>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.</p> <p>*NO (default) The registrations are not removed.</p> <p>*YES The registrations are removed.</p>
DLTTGTTBL	<p>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.</p> <p>*NO (default) The target tables are not dropped.</p> <p>*YES The target tables are dropped.</p>
RMVMBRS	<p>Specifies whether this command removes the subscription set and all the members in that subscription set.</p> <p>*NO (default) The subscription set is not removed if there are existing members in the subscription set.</p> <p>*YES The subscription set and all its subscription-set members are removed.</p>

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)
```

RMVDPRSUBM: Removing a DPR subscription-set member (iSeries)

Use the Remove DPR subscription-set member (**RMVDPRSUBM**) command to remove a single subscription-set member from a subscription set.

Table 56. *RMVDPRSUBM* command parameter definitions for *iSeries* (continued)

Parameter	Definition and prompts
RMVREG	<p>Specifies whether this command removes the registration that is associated with the target table for the subscription-set member.</p> <p>*NO (default) The registration is not removed.</p> <p>*YES The registration is removed.</p>
DLTTGTTBL	<p>Specifies whether this command drops the target table of the subscription-set member after the subscription-set member is removed.</p> <p>*NO (default) The target table is not dropped.</p> <p>*YES The target table is dropped.</p>

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/YTD TAX) RMVREG(*YES)
DLTTGTTBL(*YES)
```

RVKDPRAUT: Revoking authority (iSeries)

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.

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.

Syntax

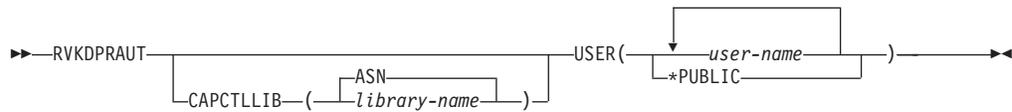


Table 57 lists the invocation parameters.

Table 57. RVKDPRAUT command parameter definitions for iSeries

Parameter	Definition and prompts
CAPCTLLIB	<p>Specifies the Capture schema, which is the name of the library under which user authority is being revoked.</p> <p>ASN (default) The Capture control tables reside in the ASN library.</p> <p><i>library-name</i> The name of the library that contains the replication control tables.</p>
USER	<p>Specifies the users whose authority is revoked. This parameter is required.</p> <p><i>user-name</i> Specifies the names of up to 50 users whose authority is revoked.</p> <p>*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.</p>

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)
```

STRDPRAPY: Starting Apply (iSeries)

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.

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.

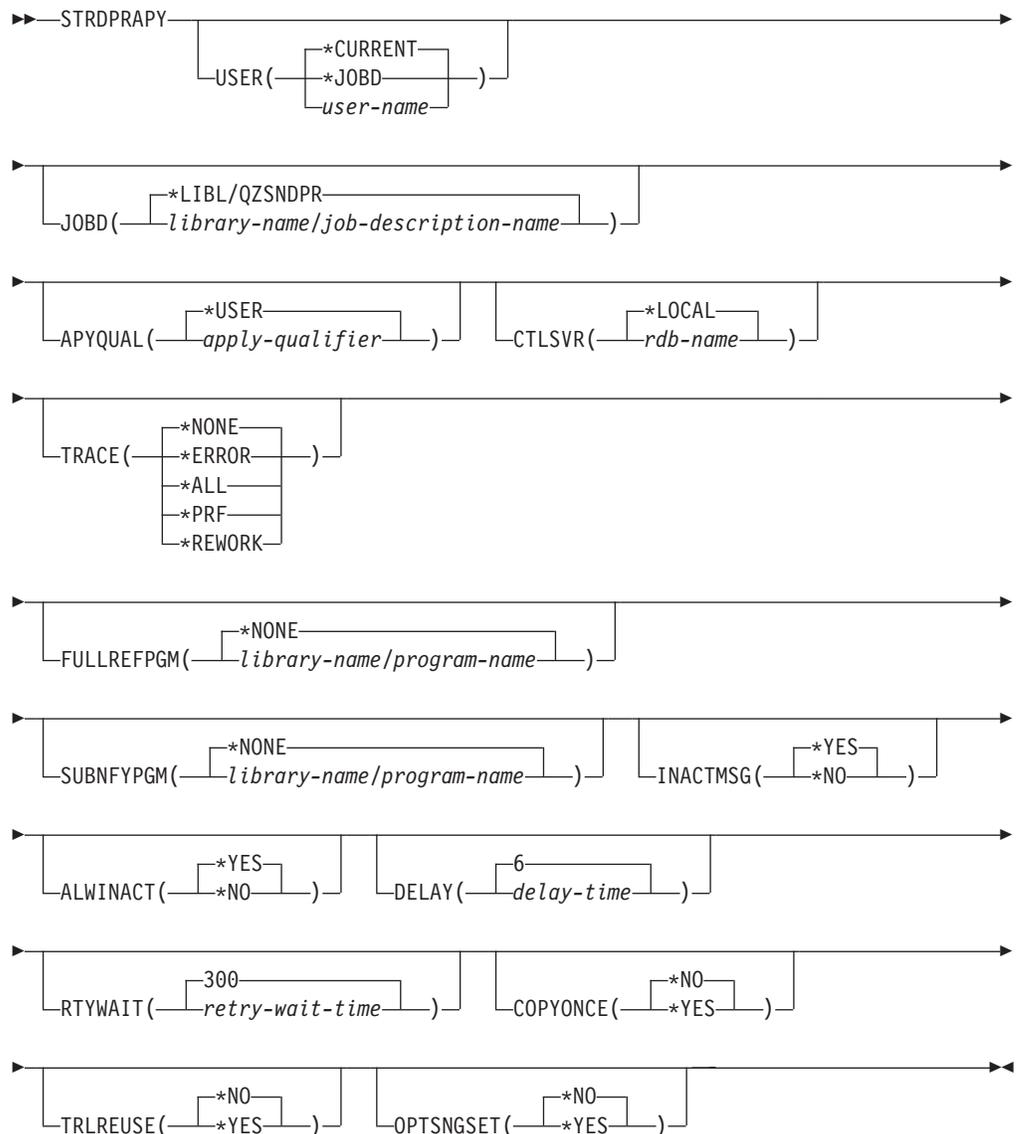


Table 58 lists the invocation parameters.

Table 58. STRDPRAPY command parameter definitions for iSeries

Parameter	Definition and prompts
USER	<p>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.</p> <p>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.</p> <p>*CURRENT (default) The user ID associated with the current job is the same user ID associated with this Apply program.</p> <p>*JOB The user ID specified in the job description associated with this Apply program. The job description cannot specify USER(*RQD).</p> <p><i>user-name</i> The user ID associated with this Apply program. The following IBM-supplied objects are <i>not</i> valid on this parameter: QDBSHR, QDFTOWN, QDOC, QLPAUTO, QLPINSTALL, QRJE, QSECOFR, QSPL, QSYS, or QTSTRQS.</p> <p>When prompting on the STRDPRAPY command, you can press the F4 key to see a list of users who defined subscription sets.</p>
JOB	<p>Specifies the name of the job description to use when submitting the Apply program.</p> <p>*LIBL/QZSNDPR (default) The default job description provided with DB2 DataPropagator for iSeries.</p> <p><i>library-name/job-description-name</i> The name of the job description used for the Apply program.</p>
APYQUAL	<p>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.</p> <p>*USER (default) The USER parameter value that you enter is used as the name of the Apply qualifier.</p> <p><i>apply-qualifier</i> 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.</p> <p>When prompting on the STRDPRAPY command, you can press the F4 key to see a list of Apply qualifier names with existing subscription sets.</p>

Table 58. STRDPRAPY command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
CTLSVR	<p>Specifies the relational database name of the system that contains the Apply control tables.</p> <p>*LOCAL (default) The Apply control tables reside locally (on the machine where you are running the STRDPRAPY command).</p> <p><i>rdb-name</i> 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.</p> <p>When prompting on the STRDPRAPY command, you can press the F4 key to see a list of available RDB names.</p>
TRACE	<p>Specifies whether the Apply program should generate a trace. The Apply program writes the trace data to a spool file called QPZSNATRC.</p> <p>*NONE (default) No trace is generated.</p> <p>*ERROR The trace contains error information only.</p> <p>*ALL The trace contains error and execution flow information.</p> <p>*PRF The trace contains information that can be used to analyze performance at different stages of the Apply program execution.</p> <p>*REWORK The trace contains information about rows that were reworked by the Apply program.</p>
FULLREFPGM	<p>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.</p> <p>When a full-refresh exit routine is used by the Apply program, the value of the ASNLOAD column in the IBMSNAP_APPLYTRAIL table is Y.</p> <p>*NONE (default) A full-refresh exit routine is not used.</p> <p><i>library-name/program-name</i> 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.</p>

Table 58. STRDPRAPY command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
SUBNFYPGM	<p>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.</p> <p>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.</p> <p>*NONE (default) An exit routine is not used.</p> <p><i>library-name/program-name</i> 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.</p>
INACTMSG	<p>Specifies whether the Apply program is to generate a message whenever the program completes its work and becomes inactive for a period of time.</p> <p>*YES (default) The Apply program generates message ASN1044 before beginning a period of inactivity. Message ASN1044 indicates how long the Apply program remains inactive.</p> <p>*NO No message is generated.</p>
ALWINACT	<p>Specifies whether the Apply program can run in an inactive (sleep) state.</p> <p>*YES (default) The Apply program sleeps if there is nothing to process.</p> <p>*NO If the Apply program has nothing to process, the job that submitted and started the Apply program ends.</p>
DELAY	<p>Specifies the delay time (in seconds) at the end of each Apply program cycle when continuous replication is used.</p> <p>6 (default) The delay time is six seconds.</p> <p><i>delay-time</i> The delay time, entered as a number between 0 and 6 inclusive.</p>
RTYWAIT	<p>Specifies how long (in seconds) the Apply program is to wait after it encounters an error before it retries the operation that failed.</p> <p>300 (default) The retry wait time is 300 seconds (five minutes).</p> <p><i>retry-wait-time</i> The wait time, entered as a number between 0 and 35000000 inclusive, before the Apply program retries the failed operation.</p>

Table 58. STRDPRAPY command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
COPYONCE	<p>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:</p> <ul style="list-style-type: none"> • (ACTIVATE > 0) in the 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. <p>The MAX_SYNCH_MINUTES limit from the IBMSNAP_SUBS_SET table and the END_OF_PERIOD timestamp from the IBMSNAP_SUBS_EVENT table are honored if specified.</p> <p>*NO (default) The Apply program does not execute one copy cycle for each eligible subscription set.</p> <p>*YES The Apply program executes one copy cycle for each eligible subscription set and then terminates.</p>
TRLREUSE	<p>Specifies whether the Apply program empties the IBMSNAP_APPLYTRAIL table when the Apply program starts.</p> <p>*NO (default) The Apply program does not empty the IBMSNAP_APPLYTRAIL table during program startup.</p> <p>*YES The Apply program empties the IBMSNAP_APPLYTRAIL table during program startup.</p>
OPTSNGSET	<p>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.</p> <p>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.</p> <p>*NO (default) The performance of the Apply program is not optimized if only one subscription set is processed.</p> <p>*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 fewer CPU resources and improving throughput rates.</p>

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 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.

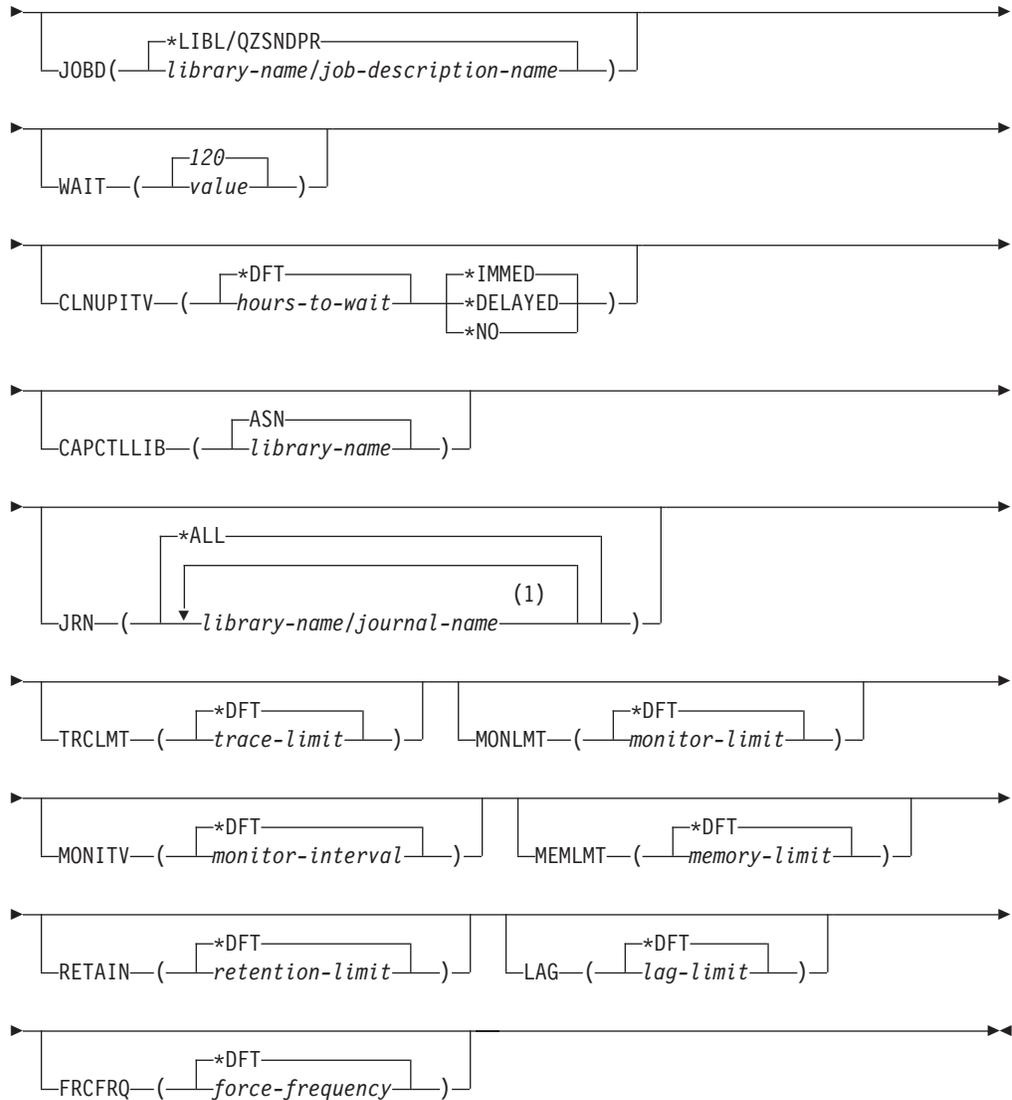
Identifying Apply program jobs

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 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.



Notes:

- 1 You can specify up to 50 journals.

Table 59 lists the invocation parameters.

Table 59. STRDPRCAP command parameter definitions for iSeries

Parameter	Definition and prompts
RESTART	<p>Specifies how the Capture program handles warm and cold starts.</p> <p>*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 <i>warm start</i> and is the normal mode of operation.</p> <p>*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 <i>cold start</i>. By specifying RESTART(*NO) and JRN(<i>library-name/journal-name</i>), you can cold start the Capture program for specified journals.</p>
JOBID	<p>Specifies the name of the job description to use when submitting the Capture program.</p> <p>*LIBL/QZSNDPR (default) Specifies the default job description provided with DB2 DataPropagator for iSeries. <i>library-name/job-description-name</i> The name of the job description used for the Capture program.</p>
WAIT	<p>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.</p> <p>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.</p> <p>120 (default) The Capture program waits 120 seconds.</p> <p><i>value</i> The maximum number of seconds that the Capture program waits.</p>

Table 59. STRDPRCAP command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
CLNUPITV	<p>Specifies the maximum amount of time (in hours) before the Capture program prunes old records from the change-data (CD), unit-of-work (UOW), IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and IBMSNAP_AUTHTKN tables.</p> <p>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.</p> <p>(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.)</p> <p>There are two parts to the CLNUPITV parameter:</p> <p>*DFT (default) The Capture program uses the value of the PRUNE_INTERVAL column from the IBMSNAP_CAPPARMS table.</p> <p><i>hours-to-wait</i> The pruning interval expressed as a specific number of hours (1 to 100).</p> <p>*IMMED (default) The Capture program prunes old records at the beginning of the specified interval (or immediately), and at each interval thereafter.</p> <p>*DELAYED The Capture program prunes old records at the end of the specified interval, and at each interval thereafter.</p> <p>*NO The Capture program does not prune records.</p>
CAPCTLLIB	<p>Specifies the Capture schema, which is the name of the library in which the Capture control tables reside.</p> <p>ASN (default) The default library in which the Capture control tables reside.</p> <p><i>library-name</i> The name of the library in which the Capture control tables reside.</p>
JRN	<p>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.</p> <p>*ALL (default) The Capture program starts working with all of the journals that have any source tables journaled to them.</p> <p><i>library-name/journal-name</i> The qualified name of the journal that you want the Capture program to work with. When entering multiple journals, separate the journals with spaces.</p>

Table 59. STRDPRCAP command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
TRCLMT	<p>Specifies the trace limit (in minutes). The Capture program prunes any IBMSNAP_CAPTRACE table rows that are older than the trace limit. The default is 10 080 minutes (seven days of trace entries).</p> <p>*DFT (default) The Capture program uses the TRACE_LIMIT column value from the IBMSNAP_CAPPARMS table.</p> <p><i>trace-limit</i> The number of minutes of trace data kept in the IBMSNAP_CAPTRACE table after pruning.</p>
MONLMT	<p>Specifies the monitor limit (in minutes). The Capture programs prunes any IBMSNAP_CAPMON table rows that are older than the monitor limit. The default is 10 080 minutes (seven days of monitor entries).</p> <p>*DFT (default) The Capture program uses the MONITOR_LIMIT column value from the IBMSNAP_CAPPARMS table.</p> <p><i>monitor-limit</i> The number of minutes of monitor data kept in the IBMSNAP_CAPMON table after pruning.</p>
MONITV	<p>Specifies how frequently (in seconds) the Capture program inserts rows into the IBMSNAP_CAPMON table. The default is 300 seconds (five minutes).</p> <p>*DFT (default) The Capture program uses the MONITOR_INTERVAL column value from the IBMSNAP_CAPPARMS table.</p> <p><i>monitor-interval</i> 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.</p>
MEMLMT	<p>Specifies the maximum size (in megabytes) of memory that the Capture journal job can use. The default is 32 megabytes.</p> <p>*DFT (default) The Capture program uses the MEMORY_LIMIT column value from the IBMSNAP_CAPPARMS table.</p> <p><i>memory-limit</i> The maximum number of megabytes for memory.</p>

Table 59. STRDPRCAP command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
RETAIN	<p>Specifies the new retention limit, which is the number of minutes that data is retained in the change-data (CD), unit-of-work (UOW), IBMSNAP_SIGNAL, and IBMSNAP_AUTHHTKN 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_AUTHHTKN data is removed if this data is older than the retention limit.</p> <p>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.</p> <p>The default is 10 080 minutes (seven days). The maximum is 3500000 minutes.</p> <p>*DFT (default) The Capture program uses the RETENTION_LIMIT column value from the IBMSNAP_CAPPARMS table.</p> <p><i>retention-limit</i> The number of minutes that the CD, UOW, IBMSNAP_SIGNAL, and IBMSNAP_AUTHHTKN data is retained.</p>
LAG	<p>Specifies the new lag limit, which is the number of minutes that the Capture program can fall behind in processing before restarting.</p> <p>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.</p> <p>The default is 10 080 minutes (seven days). The maximum is 3500000 minutes.</p> <p>*DFT (default) The Capture program uses the LAG_LIMIT column value from the IBMSNAP_CAPPARMS table.</p> <p><i>lag-limit</i> The number of minutes that the Capture program is allowed to fall behind.</p>

Table 59. STRDPRCAP command parameter definitions for iSeries (continued)

Parameter	Definition and prompts
FRCFRQ	<p>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.</p> <p>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.</p> <p>The default is 30 seconds.</p> <p>*DFT (default) The Capture program uses the COMMIT_INTERVAL column value from the IBMSNAP_CAPPARMS table.</p> <p><i>force-frequency</i> 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.</p>

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), IBMSNAP_SIGNAL, IBMSNAP_CAPMON, IBMSNAP_CAPTRACE, and 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 **JOBID** 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 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 iSeries 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 started in cold mode, whereas replication sources using a different journal are not started in cold mode.

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)
```

WRKDPRTTC: Using the DPR trace facility (iSeries)

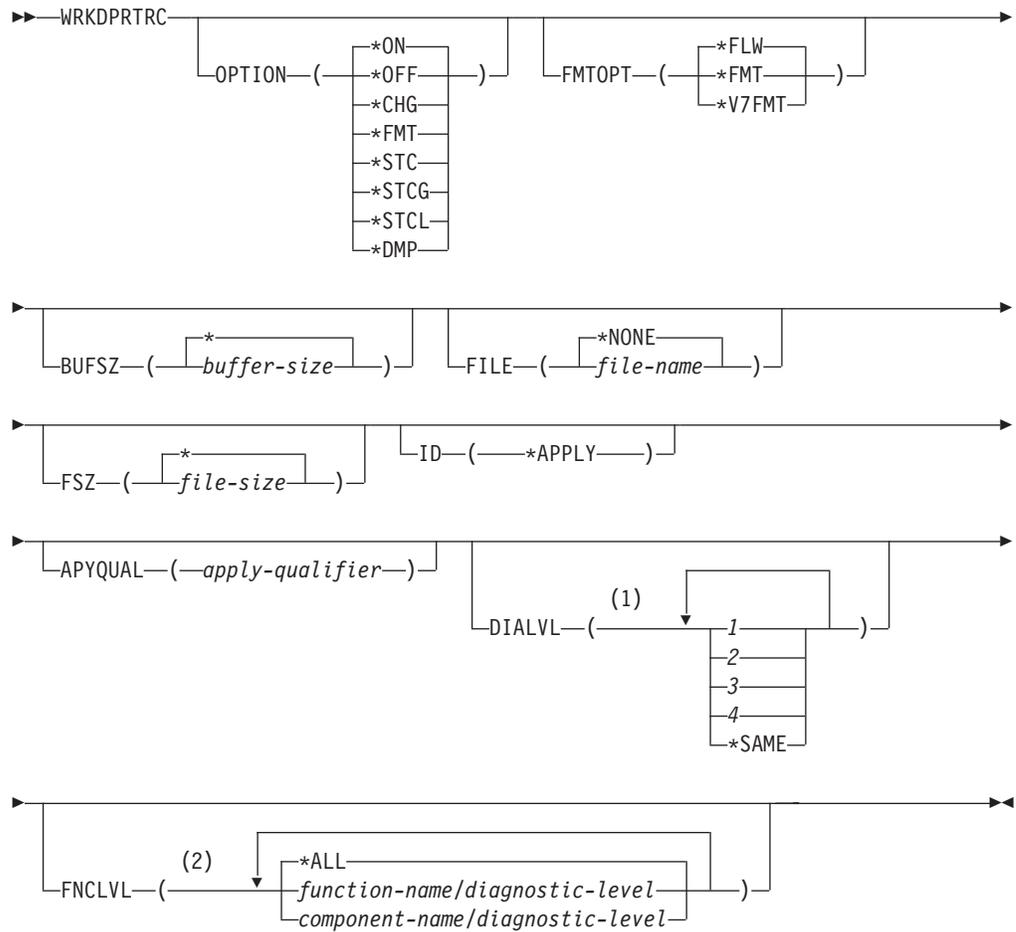
Use the DPR trace (WRKDPRTTC) command to run the trace facility to logs program flow information for specified Apply programs.

Use the DPR trace (WRKDPRTTC) 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.

Syntax



Notes:

- 1 You can specify multiple values.
- 2 You can specify up to 20 functions or components.

Table 60 lists the invocation parameters.

Table 60. WRKDPRTRC command parameter definitions for iSeries

Parameter	Definition
OPTION	<p>Specify one trace facility function.</p> <p>*ON (default) Turn the trace facility on. This option automatically creates a shared memory segment for tracing.</p> <p>*OFF Turn the trace facility off.</p> <p>*CHG Change the values of the trace facility parameters.</p> <p>*FMT Format the trace facility output from shared memory.</p> <p>*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.</p> <p>This parameter option is equivalent to the stat option of the asntrc command used on UNIX, Windows, and z/OS operating systems.</p> <p>*STCG Display the status of a trace facility in Replication Center readable format.</p> <p>*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.</p> <p>This parameter option is equivalent to the statlong option of the asntrc command used on UNIX, Windows, and z/OS operating systems.</p> <p>*DMP Write the current contents of the trace buffer to a file.</p> <p>When prompting on the WRKDPRTRC command, you can press the F4 key to see a list of trace options.</p>

Table 60. WRKDPTRRC command parameter definitions for iSeries (continued)

Parameter	Definition
FMTOPT	<p>Specifies the options of the format ID and is used with the OPTION(*FMT) parameter.</p> <p>*FLW (default) Display the flow of the function calls.</p> <p>*FMT Display the format of the trace buffer or trace file. Shows all the detailed data.</p> <p>*V7FMT Format the trace buffer or trace file information in Version 7 format.</p> <p>When prompting on the WRKDPTRRC command, you can press the F4 key to see a list of format options.</p>
BUFSZ	<p>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.</p> <p>The default is two megabytes.</p> <p>* (default) Use the two megabyte default size.</p> <p><i>buffer-size</i> The buffer size in bytes.</p>
FILE	<p>Specifies whether the trace output is written to a file.</p> <p>*NONE (default) The trace output goes to shared memory only.</p> <p><i>file-name</i> The name of the output file. If you are using the OPTION(*DMP) parameter, this file name represents the name of a dump file.</p>
FSZ	<p>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.</p> <p>The default is two gigabytes.</p> <p>* (default) Use the two gigabyte default size.</p> <p><i>file-size</i> The file size in bytes.</p>
ID	<p>Specifies the type of program to be traced.</p> <p>*APPLY (default) An Apply program trace.</p>
APYQUAL	<p>Specifies the name of Apply program to be traced.</p> <p><i>apply-qualifier</i> The name of the Apply qualifier.</p>

Table 60. WRKDPTRTC command parameter definitions for iSeries (continued)

Parameter	Definition
DIALVL	<p>Specifies the types of trace records to be recorded by the trace facility. Trace records are categorized by a diagnostic mask number:</p> <ul style="list-style-type: none"> 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. <p>*SAME This command uses the diagnostic level settings from the previous trace facility.</p> <p>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.</p> <p>Important: The number levels are <i>not</i> inclusive.</p> <p>When you start the trace facility, the default is DIALVL(1234). When you subsequently invoke the trace facility, the default is *SAME.</p> <p>When prompting on the WRKDPTRTC command, you can press the F4 key to see a list of available diagnostic levels.</p>
FNCLVL	<p>Specifies if a particular function or component identifier is to be traced.</p> <p>*ALL (default) All functions and components are included in the trace facility.</p> <p><i>function-name/diagnostic-level</i> The name of the function to be traced and the corresponding diagnostic mask numbers.</p> <p><i>component-name/diagnostic-level</i> The name of the component to be traced and the corresponding diagnostic mask numbers.</p> <p>You can enter up to 20 function or component names.</p>

Examples for WRKDPTRTC

The following examples illustrate how to use the WRKDPTRTC 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:

```
WRKDPTRTC OPTION(*ON) FILE(TRCFILE) ID(*APPLY) APYQUAL(AQ1)
```

Example 2:

To end an Apply trace on the Apply qualifier AQ1:

```
WRKDPTRTC 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:

```
WRKDPTRTC OPTION(*CHG) ID(*APPLY) APYQUAL(AQ1) DIALVL(34)
```

Example 4:

To display the status of an Apply trace on the Apply qualifier AQ1:

```
WRKDPTRTC OPTION(*STC) ID(*APPLY) APYQUAL(AQ1)
```

Example 5:

To display the function calls on the Apply qualifier AQ1 at diagnostic levels 3 and 4:

```
WRKDPTRTC 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:

```
WRKDPTRTC OPTION(*DMP) FILE(DMPFILE) ID(*APPLY) APYQUAL(AQ1)
```

Chapter 24. SQL replication table structures

Relational database tables are used to store information for the replication program at each server: Capture control server, Apply control server, Monitor control server, and target server. These tables are called *control tables*.

The following topics describe the structure of the control tables and other tables that are specific to SQL replication.

Tables at a glance

The following diagrams can be used for quick reference to the control tables for the Capture control server, Apply control server, and Monitor control server.

Figure 7 on page 374, Figure 8 on page 375, and Figure 9 on page 376 show the tables at the Capture control server, the columns in each table, and the indexes on each table. Figure 10 on page 377 and Figure 11 on page 378 show the tables at the Apply control server, the columns in each table, and the indexes on each table. Figure 12 on page 379 and Figure 13 on page 380 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 2)

<p><i>OS/400 only</i></p> <p>schema.IBMSNAP_AUTHTKN (no unique index)</p> <p>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</p>	<p>ASN.IBMSNAP_CAPSCHEMAS (CAP_SCHEMA_NAME)</p> <p>CAP_SCHEMA_NAME VARCHAR(30)</p> <p><i>OS/400</i></p> <p>ASN.IBMSNAP_CAPSCHEMAS (CAP_SCHEMA_NAME)</p> <p>CAP_SCHEMA_NAME VARCHAR(30) STATUS CHAR(1)</p>
<p><i>UNIX, Windows, and z/OS only</i></p> <p>schema.IBMSNAP_CAPENQ (no unique index)</p> <p>LOCKNAME CHAR(9)</p>	<p>schema.IBMSNAP_CAPTRACE (TRACE_TIME)</p> <p>OPERATION CHAR(8) NOT NULL TRACE_TIME TIMESTAMP NOT NULL DESCRIPTION VARCHAR(1024) NOT NULL</p> <p><i>OS/400</i></p> <p>schema.IBMSNAP_CAPTRACE (TRACE_TIME)</p> <p>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</p>
<p>schema.IBMSNAP_CAPMON (MONITOR_TIME)</p> <p>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_TRAN_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</p>	<p>schema.IBMSNAP_PRUNCNTL (SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL, APPLY_QUAL, SET_NAME, TARGET_SERVER, TARGET_TABLE, TARGET_OWNER, MAP_ID)</p> <p>TARGET_SERVER CHAR(18) NOT NULL TARGET_OWNER VARCHAR(30) NOT NULL TARGET_TABLE VARCHAR(128) NOT NULL SYNCHTIME TIMESTAMP SYNCHPOINT CHAR(10) FOR BIT DATA SOURCE_OWNER VARCHAR(30) NOT NULL SOURCE_TABLE VARCHAR(128) 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) PHYS_CHANGE_TABLE VARCHAR(128) MAP_ID VARCHAR(10) NOT NULL</p>
<p>schema.IBMSNAP_CAPPARMS (no unique index)</p> <p>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) LOGSDOUT CHAR(1) SLEEPINTERVAL SMALLINT CAPTURE_PATH VARCHAR(1040) STARTMODE VARCHAR(10)</p>	<p>schema.IBMSNAP_PRUNE_LOCK (no unique index)</p> <p>DUMMY CHAR(1)</p>

Figure 7. 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 2)

<p>schema.IBMSNAP_PRUNE_SET (TARGET_SERVER, APPLY_QUAL, SET_NAME)</p> <p>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</p>	<p>schema.IBMSNAP_REG_SYNCH (TRIGGER_ME)</p> <p>TRIGGER_ME CHAR(1) NOT NULL</p>
<p>schema.IBMSNAP_REGISTER (SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL)</p> <p>SOURCE_OWNER VARCHAR(30) NOT NULL SOURCE_TABLE VARCHAR(128) 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) CD_TABLE VARCHAR(128) PHYS_CHANGE_OWNER VARCHAR(30) PHYS_CHANGE_TABLE VARCHAR(128) 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) CCD_TABLE VARCHAR(128) 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) STATE CHAR(1) STATE_INFO CHAR(8)</p>	<p>schema.IBMSNAP_RESTART (no unique index)</p> <p>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</p> <p><i>OS/400</i></p> <p>schema.IBMSNAP_RESTART (JRN_LIB, JRN_NAME)</p> <p>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</p> <p>UID INTEGER NOT NULL SEQNBR BIGINT NOT NULL JRN_LIB CHAR(10) NOT NULL JRN_NAME CHAR(10) NOT NULL STATUS CHAR(1)</p>
<p><i>OS/400 only</i></p> <p>schema.IBMSNAP_REG_EXT (VERSION, SOURCE_OWNER, SOURCE_TABLE, SOURCE_VIEW_QUAL)</p> <p>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</p>	<p>schema.IBMSNAP_SEQTABLE (SEQ)</p> <p>SEQ INTEGER NOT NULL</p>
	<p>schema.IBMSNAP_SIGNAL (SIGNAL_TIME)</p> <p>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</p>
	<p>schema.IBMSNAP_UOW (IBMSNAP_COMMITSEQ, IBMSNAP_LOGMARKER)</p> <p>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) NOT NULL IBMSNAP_REJ_CODE CHAR(1) NOT NULL WITH DEFAULT IBMSNAP_APPLY_QUAL CHAR(18) NOT NULL WITH DEFAULT</p>

Figure 8. 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

<i>UNIX, Windows, and z/OS only</i> (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
<i>OS/400 only</i> (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_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

schema.IBMSNAP_SEQTABLE

<i>Informix only</i> (SEQ)	
SEQ	INTEGER 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.

Figure 9. 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 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) NOT NULL
TARGET_TABLE	VARCHAR(128) 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_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)

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_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) NOT NULL
TGT_CAPTURE_SCHEMA	VARCHAR(30)
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_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) NOT NULL
SOURCE_TABLE	VARCHAR(128) NOT NULL
SOURCE_VIEW_QUAL	SMALLINT NOT NULL
TARGET_OWNER	VARCHAR(30) NOT NULL
TARGET_TABLE	VARCHAR(128) 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)
LOADX_SRC_N_TABLE	VARCHAR(128)

Figure 11. 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

ASNJBMSNAP_ALERTS (MONITOR_QUAL, COMPONENT, SERVER_NAME, SCHEMA_OR_QUAL, 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) 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

ASNJBMSNAP_GROUPS (GROUP_NAME)	
GROUP_NAME	VARCHAR(127) NOT NULL
DESCRIPTION	VARCHAR(1024)

ASNJBMSNAP_MONENQ (MONITOR_QUAL)	
MONITOR_QUAL	CHAR(18) NOT NULL

ASNJBMSNAP_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

ASNJBMSNAP_CONDITIONS (MONITOR_QUAL, SERVER_NAME, COMPONENT, SCHEMA_OR_QUAL, SET_NAME, CONDITION_NAME)	
SERVER_NAME	CHAR(18) NOT NULL
COMPONENT	CHAR(1) NOT NULL
SCHEMA_OR_QUAL	VARCHAR(30) NOT NULL
SET_NAME	CHAR(18) NOT NULL WITH DEFAULT
MONITOR_QUAL	CHAR(18) NOT NULL
SERVER_ALIAS	CHAR(8)
ENABLED	CHAR(1) NOT NULL
CONDITION_NAME	CHAR(18) NOT NULL
PARAM_INT	INT
PARAM_CHAR	VARCHAR(128)
CONTACT_TYPE	CHAR(1) NOT NULL
CONTACT	VARCHAR(127) NOT NULL

ASNJBMSNAP_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

ASNJBMSNAP_CONTACTGRP (GROUP_NAME, CONTACT_NAME)	
GROUP_NAME	VARCHAR(127) NOT NULL
CONTACT_NAME	VARCHAR(127) NOT NULL

ASNJBMSNAP_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)

ASNJBMSNAP_MONTRAIL (no unique 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 12. 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 13. 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.

Tables 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 ASNCLP command-line program or Replication Center. For iSeries, 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 iSeries to create Capture control tables in alternate capture schemas.

Table 61 describes the control tables at the Capture server.

Table 61. Quick reference for tables used at the Capture control server

Table name	Description
"IBMSNAP_CAPSCHEMAS table" on page 387	Contains the names of all Capture schemas
IBMSNAP_AUTHTKN table (iSeries)	Contains information to support update-anywhere replication.
"IBMSNAP_CAPENQ table (Linux, UNIX, Windows, z/OS)" on page 382	For each Capture schema, this table is used to ensure that: <ul style="list-style-type: none"> • 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.
"CD table" on page 391	Contains information about changes that occur at the source. This table is not created until you register a replication source.

Table 61. Quick reference for tables used at the Capture control server (continued)

Table name	Description
"CCD table (non-DB2)" on page 390	Contains information about changes that occur at the source and additional columns to identify the sequential ordering of those changes.
"IBMSNAP_CAPMON table" on page 382	Contains operational statistics that help monitor the progress of the Capture program.
"IBMSNAP_CAPPARMS table" on page 384	Contains parameters that you can specify to control the operations of the Capture program.
"IBMSNAP_CAPTRACE table" on page 389	Contains messages from the Capture program.
"IBMSNAP_PARTITIONINFO table" on page 392	Contains information that enables the Capture program to restart from the earliest required log sequence number.
"IBMSNAP_PRUNE_LOCK table" on page 395	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).
"IBMSNAP_PRUNE_SET table" on page 396	Coordinates the pruning of CD tables.
"IBMSNAP_PRUNCNTL table" on page 393	Coordinates synchpoint updates between the Capture and Apply programs.
IBMSNAP_REG_EXT (iSeries)	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.
"IBMSNAP_REGISTER table" on page 398	Contains information about replication sources, such as the names of the replication source tables, their attributes, and their corresponding CD and CCD table names.
"IBMSNAP_REG_SYNCH table (non-DB2 relational)" on page 404	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.
"IBMSNAP_RESTART table" on page 405	Contains information that enables the Capture program to resume capturing from the correct point in the log or journal. For iSeries environments, this table is also used to determine the starting time of the RCVJRNE (Receive Journal Entry) command.
"IBMSNAP_SEQTABLE table (Informix)" on page 407	Contains a sequence of unique numbers that SQL replication uses as the equivalent of log sequence numbers for Informix tables.
"IBMSNAP_SIGNAL table" on page 407	Contains all signals used to prompt the Capture program. These signals can be sent manually or by the Apply program.
"IBMSNAP_UOW table" on page 410	Provides additional information about transactions that have been committed to a source table.

IBMSNAP_CAPENQ table (Linux, UNIX, Windows, z/OS)

For a single Capture schema, the IBMSNAP_CAPENQ table ensures that only one Capture program is running per database, subsystem, or data-sharing group.

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 iSeries servers.

While running, the Capture program exclusively locks this table.

Table 62 provides a brief description of the column in the IBMSNAP_CAPENQ table.

Table 62. Column in the IBMSNAP_CAPENQ table

Column name	Description
LOCKNAME	Data type: CHAR(9); Nullable: Yes
	This column contains no data.

IBMSNAP_CAPMON table

The Capture program inserts a row in the IBMSNAP_CAPMON table after each interval to provide you with operational statistics. The Replication Center uses information in this table (and in other tables) so that you can monitor the status of the Capture program.

Server: Capture control server

Default schema: ASN

Index: MONITOR_TIME

In the 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 63 provides a brief description of the columns in the IBMSNAP_CAPMON table.

Table 63. Columns in the IBMSNAP_CAPMON 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.

Table 63. Columns in the IBMSNAP_CAPMON table (continued)

Column name	Description
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.
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 (iSeries)	Data type: CHAR(10); Nullable: Yes The library name of the journal that the Capture program was processing.
JRN_NAME (iSeries)	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.

Table 63. Columns in the IBMSNAP_CAPMON table (continued)

Column name	Description
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.

IBMSNAP_CAPPARMS table

The IBMSNAP_CAPPARMS 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.

Server: Capture control server

Default schema: ASN

Index: None

This table contains information that you can update by using SQL.

Table 64 provides a brief description of the columns in the IBMSNAP_CAPPARMS table.

Table 64. Columns in the IBMSNAP_CAPPARMS 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.

Table 64. Columns in the IBMSNAP_CAPPARMS table (continued)

Column name	Description
PRUNE_INTERVAL	<p>Data type: INT; Nullable: Yes</p> <p>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.</p>
TRACE_LIMIT	<p>Data type: INT; Nullable: Yes</p> <p>The number of minutes that rows remain in the 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.</p>
MONITOR_LIMIT	<p>Data type: INT; Nullable: Yes</p> <p>The number of minutes that rows remain in the 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.</p>
MONITOR_INTERVAL	<p>Data type: INT; Nullable: Yes</p> <p>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.</p>
MEMORY_LIMIT	<p>Data type: SMALLINT; Nullable: Yes</p> <p>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.</p>
REMOTE_SRC_SERVER	<p>Data type: CHAR(18); Nullable: Yes</p> <p>Reserved for future options of SQL replication. Currently this column contains the default value of null.</p>
AUTOPRUNE	<p>Data type: CHAR(1); Nullable: Yes</p> <p>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:</p> <p>Y Autopruning is on. N Autopruning is off.</p>
TERM	<p>Data type: CHAR(1); Nullable: Yes</p> <p>A flag that indicates whether the Capture program terminates when DB2 terminates:</p> <p>Y The Capture program terminates when DB2 terminates. N The Capture program stays active and waits for DB2 to be restarted.</p>

Table 64. Columns in the IBMSNAP_CAPPARMS table (continued)

Column name	Description
AUTOSTOP	<p>Data type: CHAR(1); Nullable: Yes</p> <p>A flag that indicates whether the Capture program stops capturing changes as soon as it reaches the end of the active logs:</p> <p>Y The Capture program stops as soon as it reaches the end of the active logs.</p> <p>N The Capture program continues running when it reaches the end of the active logs.</p>
LOGREUSE	<p>Data type: CHAR(1); Nullable: Yes</p> <p>A flag that indicates whether the Capture program overwrites the Capture log file or appends to it.</p> <p>Y The Capture program reuses the log file by first deleting it and then recreating it when the Capture program is restarted.</p> <p>N The Capture program appends new information to the Capture log file.</p>
LOGSTDOUT	<p>Data type: CHAR(1); Nullable: Yes</p> <p>A flag that indicates where the Capture program directs the log file messages:</p> <p>Y The Capture program directs log file messages to both the standard out (STDOUT) and the log file.</p> <p>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.</p>
SLEEP_INTERVAL (Linux, UNIX, Windows, z/OS)	<p>Data type: SMALLINT; Nullable: Yes</p> <p>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).</p>
CAPTURE_PATH	<p>Data type: VARCHAR(1040); Nullable: Yes</p> <p>The path where the output from the Capture program is sent.</p>

Table 64. Columns in the IBMSNAP_CAPPARMS table (continued)

Column name	Description
STARTMODE	<p>Data type: VARCHAR(10); Nullable: Yes</p> <p>The processing procedure that the Capture program uses when it is started:</p> <p>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.</p> <p>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.</p> <p>warmns The Capture program warm starts. If it can't warm start, it does <i>not</i> 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.</p>

IBMSNAP_CAPSCHEMAS table

The IBMSNAP_CAPSCHEMAS table holds the names of all Capture schemas. It allows the administration tools 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.

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 following two tables show operating system-specific layouts of the IBMSNAP_CAPSCHEMAS table.

Table 65. Columns in the IBMSNAP_CAPSCHEMAS table for all operating systems other than iSeries

Column name	Description
CAP_SCHEMA_NAME	<p>Data type: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode; Nullable: Yes</p> <p>The name of a Capture schema. A row exists for each Capture schema.</p>

Table 66. Columns in the Capture schemas table for iSeries

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.

IBMSNAP_AUTHTKN table (iSeries)

The IBMSNAP_AUTHTKN table is used in the iSeries 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. The Capture program prunes this table based on the retention limit that you set.

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.

Table 67 provides a brief description of the columns in the IBMSNAP_AUTHTKN table.

Table 67. Columns in the IBMSNAP_AUTHTKN 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.

Table 67. Columns in the IBMSNAP_AUTHTKN table (continued)

Column name	Description
IBMSNAP_LOGMARKER	Data type: TIMESTAMP; Nullable: No The approximate time that the transaction was committed at the Capture control server.

IBMSNAP_CAPTRACE table

The Capture trace table contains messages from the Capture program.

Server: Capture control server

Default schema: ASN

Index: TRACE_TIME

The following two tables show operating system-specific layouts of the IBMSNAP_CAPTRACE table.

Table 68. Columns in the IBMSNAP_CAPTRACE 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.
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 69. Columns in the Capture trace table for iSeries

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.

Table 69. Columns in the Capture trace table for iSeries (continued)

Column name	Description
JOB_NAME	<p>Data type: CHAR(26); Nullable: No</p> <p>The fully qualified name of the job that wrote this trace entry.</p> <p>Position</p> <p>Description</p> <p>1–10 The Capture schema name or the journal job name</p> <p>11–20 The ID of the user who started the Capture program</p> <p>21–26 The job number</p>
JOB_STR_TIME	<p>Data type: TIMESTAMP; Nullable: No</p> <p>The starting time of the job that is named in the JOB_NAME column.</p>
DESCRIPTION	<p>Data type: VARCHAR(298); Nullable: No</p> <p>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.</p>

CCD table (non-DB2)

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.

Server: Capture control server

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause a loss of data.

The Capture control server 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 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 SQL 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.

Table 70 on page 391 provides a brief description of the columns in the CCD table.

Table 70. 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.
<i>user key columns</i>	If the CCD table is condensed, this column contains the columns that make up the target key.
<i>user non-key columns</i>	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.
<i>user computed columns</i>	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.

CD table

Change-data (CD) tables record all committed changes made to a replication source. Pruning of the CD table is coordinated by the IBMSNAP_PRUNE_SET table. 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.

Server: Capture control server

Important: Use caution when you update this table using SQL. Altering this table inappropriately can cause a loss of data.

Table 71 provides a list and a brief description of the columns in the CD table.

Table 71. 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 processed 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

Table 71. Columns in the CD table (continued)

Column name	Description
<i>user column after-image</i>	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.
<i>user column before-image</i>	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 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.

IBMSNAP_PARTITIONINFO table

The IBMSNAP_PARTITIONINFO table augments the 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.

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.

In a multi-partitioned environment, the IBMSNAP_PARTITIONINFO table and the IBMSNAP_RESTART table replace the IBMSNAP_WARM_START table from SQL 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 72 provides a brief description of the columns in the IBMSNAP_PARTITIONINFO table.

Table 72. Columns in the IBMSNAP_PARTITIONINFO table

Column name	Description
PARTITIONID	Data type: INT; Nullable: No The partition ID for each valid partition.

Table 72. Columns in the IBMSNAP_PARTITIONINFO table (continued)

Column name	Description
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.

IBMSNAP_PRUNCNTL table

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 IBMSNAP_PRUNE_SET table during pruning. It is also used during the initialization handshake process between the Apply and Capture programs.

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.

For DB2 sources, you can invoke pruning by issuing the prune command or have it done automatically. For non-DB2 relational sources, pruning is done by the pruning trigger that was created when you registered the source.

Table 73 provides a brief description of the columns in the IBMSNAP_PRUNCNTL table.

Table 73. Columns in the IBMSNAP_PRUNCNTL 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.

Table 73. Columns in the IBMSNAP_PRUNCNTL table (continued)

Column name	Description
TARGET_TABLE	<p>Data type: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; Nullable: No</p> <p>The name of the target table or view for this member.</p>
SYNCHTIME	<p>Data type: TIMESTAMP; Nullable: Yes</p> <p>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.</p>
SYNCHPOINT	<p>Data type: CHAR(10) for bit data; Nullable: Yes</p> <p>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.</p>
SOURCE_OWNER	<p>Data type: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode; Nullable: No</p> <p>The high-level qualifier of the source table or view for this member.</p>
SOURCE_TABLE	<p>Data type: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; Nullable: No</p> <p>The name of the source table or view for this member.</p>
SOURCE_VIEW_QUAL	<p>Data type: SMALLINT; Nullable: No</p> <p>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.</p>
APPLY_QUAL	<p>Data type: CHAR(18); Nullable: No</p> <p>The Apply qualifier that identifies which Apply program is processing this member.</p>
SET_NAME	<p>Data type: CHAR(18); Nullable: No</p> <p>The name of the subscription set that this subscription-set member belongs to.</p>
CNTL_SERVER	<p>Data type: CHAR(18); Nullable: No</p> <p>The name of the server where the Apply control tables reside for this Apply program, which is identified by the APPLY_QUAL.</p>

Table 73. Columns in the IBMSNAP_PRUNCNTL table (continued)

Column name	Description														
TARGET_STRUCTURE	<p>Data type: SMALLINT; Nullable: No</p> <p>A value that identifies the type of target table or view:</p> <table> <tr> <td>1</td> <td>Source table</td> </tr> <tr> <td>3</td> <td>CCD table</td> </tr> <tr> <td>4</td> <td>Point-in-time table</td> </tr> <tr> <td>5</td> <td>Base aggregate table</td> </tr> <tr> <td>6</td> <td>Change aggregate table</td> </tr> <tr> <td>7</td> <td>Replica table</td> </tr> <tr> <td>8</td> <td>User copy table</td> </tr> </table>	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
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	<p>Data type: CHAR(8); Nullable: Yes</p> <p>The DB2 alias corresponding to the Apply control server named in the CNTL_SERVER column.</p>														
PHYS_CHANGE_OWNER	<p>Data type: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode; Nullable: Yes</p> <p>The value in the PHYS_CHANGE_OWNER column from the IBMSNAP_REGISTER table that is associated with the source of this particular subscription-set member.</p>														
PHYS_CHANGE_TABLE	<p>Data type: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; Nullable: Yes</p> <p>The value in the PHYS_CHANGE_TABLE column from the IBMSNAP_REGISTER table that is associated with the source of this particular subscription-set member.</p>														
MAP_ID	<p>Data type: VARCHAR(10); Nullable: No</p> <p>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.</p>														

IBMSNAP_PRUNE_LOCK table

The IBMSNAP_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.

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.

IBMSNAP_PRUNE_SET table

The IBMSNAP_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 IBMSNAP_PRUNCNTL table, which has one row for each source-to-target mapping, the IBMSNAP_PRUNE_SET table has one row for each subscription 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.

Table 74 provides a brief description of the columns in the IBMSNAP_PRUNE_SET table.

Table 74. Columns in the IBMSNAP_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.

IBMSNAP_REG_EXT (iSeries)

The IBMSNAP_REG_EXT table is an iSeries-specific table that provides supplemental information for the IBMSNAP_REGISTER table. For every row in the IBMSNAP_REGISTER table, there is a matching row in the IBMSNAP_REG_EXT table that contains additional iSeries-specific columns.

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.

This table is maintained by a trigger program (program QZSNJLV8 in library QDP4) on the IBMSNAP_REGISTER table. The trigger is defined at the time the IBMSNAP_REGISTER table is created.

The information from this table is used to track where and how you defined your replication sources on an iSeries server.

Table 75 provides a brief description of the columns in the IBMSNAP_REG_EXT table.

Table 75. Columns in the IBMSNAP_REG_EXT 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 75. Columns in the IBMSNAP_REG_EXT table (continued)

Column name	Description
CMT_BEHAVIOR_CASE	<p>Data type: SMALLINT; Nullable: No, with default; Default: 0</p> <p>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.</p> <p>-1 The commitment control pattern is not yet established for the applications. This is the initial value in the column.</p> <p>0 None of the applications that update the source uses commitment control.</p> <p>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.</p> <p>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.</p>
MAX_ROWS_BTWN_CMTS	<p>Data type: SMALLINT; Nullable: No, with default; Default: 0</p> <p>The maximum number of rows that the Capture program can process before it commits data to the CD table.</p>

IBMSNAP_REGISTER table

The IBMSNAP_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.

Server: Capture control server

Default schema: ASN

Index: 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 table is the place you should look if you need to know how you defined your replication sources.

Table 76 provides a brief description of the columns in the IBMSNAP_REGISTER table.

Table 76. Columns in the IBMSNAP_REGISTER table

Column name	Description
SOURCE_OWNER	<p>Data type: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; Nullable: No</p> <p>The high-level qualifier of the source table or view that you registered.</p>

Table 76. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description														
SOURCE_TABLE	<p>Data type: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; Nullable: No</p> <p>The name of the source table or view that you registered.</p>														
SOURCE_VIEW_QUAL	<p>Data type: SMALLINT; Nullable: No</p> <p>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.</p>														
GLOBAL_RECORD	<p>Data type: CHAR(1); Nullable: No</p>														
SOURCE_STRUCTURE	<p>Data type: SMALLINT; Nullable: No</p> <p>A value that identifies the structure of the source table or view:</p> <table border="0"> <tr> <td>1</td> <td>User table</td> </tr> <tr> <td>3</td> <td>CCD table</td> </tr> <tr> <td>4</td> <td>Point-in-time table</td> </tr> <tr> <td>5</td> <td>Base aggregate table</td> </tr> <tr> <td>6</td> <td>Change aggregate table</td> </tr> <tr> <td>7</td> <td>Replica table</td> </tr> <tr> <td>8</td> <td>User copy table</td> </tr> </table>	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
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	<p>Data type: CHAR(1); Nullable: No</p> <p>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:</p> <table border="0"> <tr> <td>Y</td> <td>The source is condensed.</td> </tr> <tr> <td>N</td> <td>The source is not condensed.</td> </tr> <tr> <td>A</td> <td>The source is a base-aggregate or change-aggregate table.</td> </tr> </table>	Y	The source is condensed.	N	The source is not condensed.	A	The source is a base-aggregate or change-aggregate table.								
Y	The source is condensed.														
N	The source is not condensed.														
A	The source is a base-aggregate or change-aggregate table.														
SOURCE_COMPLETE	<p>Data type: CHAR(1); Nullable: No</p> <p>A flag that indicates how the source table stores rows of primary key values:</p> <table border="0"> <tr> <td>Y</td> <td>The source table contains a row for every primary key value of interest.</td> </tr> <tr> <td>N</td> <td>The source table contains a subset of rows of primary key values.</td> </tr> </table>	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.										
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	<p>Data type: VARCHAR(30), DB2 UDB for z/OS Version 8 new-function mode subsystems; Nullable: Yes</p> <p>The high-level qualifier of the source's CD table.</p> <p>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.</p> <p>For views as sources This column contains the high-level qualifier of the CD view.</p> <p>For external CCD tables as sources This column is null.</p>														

Table 76. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
CD_TABLE	<p>Data type: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; Nullable: Yes</p> <p>The name of the source's CD table.</p> <p>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.</p> <p>For views as sources This column contains the name of the CD view.</p> <p>For external CCD tables as sources This column is null.</p>
PHYS_CHANGE_OWNER	<p>Data type: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; Nullable: Yes</p> <p>The high-level qualifier of the table or view that the Apply program uses for change-capture replication:</p> <p>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.</p> <p>For views as sources This column contains the high-level qualifier of the physical CD table that is associated with that source view.</p> <p>For external CCD tables as sources This column contains the high-level qualifier of the external CCD table.</p>
PHYS_CHANGE_TABLE	<p>Data type: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; Nullable: Yes</p> <p>The name of the table or view that the Apply program uses for change-capture replication:</p> <p>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.</p> <p>For views as sources This column contains the name of the physical CD table that is associated with that source view.</p> <p>For external CCDs as sources This column contains the name of the external CCD table.</p>
CD_OLD_SYNCHPOINT	<p>Data type: CHAR(10) for bit data; Nullable: Yes</p> <p>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.</p>
CD_NEW_SYNCHPOINT	<p>Data type: CHAR(10) for bit data; Nullable: Yes</p> <p>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.</p>

Table 76. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
DISABLE_REFRESH	<p>Data type: SMALLINT; Nullable: Yes</p> <p>A flag that indicates whether full refreshes are allowed:</p> <p>0 Full refreshes are allowed.</p> <p>1 Full refreshes are prevented.</p>
CCD_OWNER	<p>Data type: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; Nullable: Yes</p> <p>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.</p>
CCD_TABLE	<p>Data type: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; Nullable: Yes</p> <p>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.</p>
CCD_OLD_SYNCHPOINT	<p>Data type: CHAR(10) for bit data; Nullable: Yes</p> <p>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.</p>
SYNCHPOINT	<p>Data type: CHAR(10) for bit data; Nullable: Yes</p> <p>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.</p>
SYNCHTIME	<p>Data type: TIMESTAMP; Nullable: Yes</p> <p>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.</p>
CCD_CONDENSED	<p>Data type: CHAR(1); Nullable: Yes</p> <p>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:</p> <p>Y The internal CCD is condensed.</p> <p>N The internal CCD is not condensed.</p> <p>NULL No internal CCD table is defined for this source.</p>

Table 76. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
CCD_COMPLETE	<p>Data type: CHAR(1); Nullable: Yes</p> <p>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:</p> <p>N The internal CCD is not complete.</p> <p>NULL No internal CCD table is defined for this source.</p>
ARCH_LEVEL	<p>Data type: CHAR(4); Nullable: No</p> <p>The architectural level of the replication control tables:</p> <p>0801 Version 8 SQL Replication</p> <p>0803 Version 8 SQL Replication with enhanced support for Oracle sources</p> <p>0805 Version 8 SQL Replication with support for DB2 for z/OS new-function mode</p>
DESCRIPTION	<p>Data type: CHAR(254); Nullable: Yes</p> <p>A description of the replication source.</p>
BEFORE_IMG_PREFIX	<p>Data type: VARCHAR(4); Nullable: Yes</p> <p>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:</p> <p>1 For an ASCII or an EBCDIC single byte prefix character.</p> <p>2 For an ASCII double byte prefix character.</p> <p>4 For an EBCDIC DBCS prefix character. This length allows for shift-in and shift-out characters.</p>
CONFLICT_LEVEL	<p>Data type: CHAR(1); Nullable: Yes</p> <p>A flag that indicates the level of conflict detection for this source:</p> <p>0 The Apply program does not check for conflicts. Data consistency must be enforced by your application to avoid potential conflicting updates.</p> <p>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.</p> <p>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.</p>

Table 76. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
CHG_UPD_TO_DEL_INS	<p>Data type: CHAR(1); Nullable: Yes</p> <p>A flag that indicates how the Capture program stores updates in the CD table.</p> <p>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.</p> <p>N Each update to the source table is stored in a single row in the CD table.</p>
CHGONLY	<p>Data type: CHAR(1); Nullable: Yes</p> <p>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.</p> <p>Y The Capture program only captures changes that occur in registered columns in the source table.</p> <p>N The Capture program captures changes from all columns in the source table.</p>
RECAPTURE	<p>Data type: CHAR(1); Nullable: Yes</p> <p>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.</p> <p>For tables at the master site:</p> <p>N Updates to the master that were applied from a replica are not recaptured and will not be replicated to other replicas.</p> <p>Y Updates to the master that were applied from a replica and will be replicated to other replicas.</p> <p>For tables at a replica site:</p> <p>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.</p> <p>N Updates to the replica that were applied from the master are not recaptured.</p>
OPTION_FLAGS	<p>Data type: CHAR(4); Nullable: No</p> <p>Reserved for future options of SQL replication. Currently this column contains the default value of NNNN.</p>

Table 76. Columns in the IBMSNAP_REGISTER table (continued)

Column name	Description
STOP_ON_ERROR	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: Y.</p> <p>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:</p> <p>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.</p> <p>N The Capture program stops the registration but does not terminate when an error occurs while it is trying to start, reinitialize, or insert a row into the CD table; it continues to process other registrations.</p>
STATE	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: I.</p> <p>A flag that indicates what state the registration is in:</p> <p>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.</p> <p>A The registration is active.</p> <p>I The registration is inactive.</p>
STATE_INFO	<p>Data type: CHAR(8); Nullable: Yes;</p> <p>If the Capture program stopped processing the registration, this column contains the error message that was issued regarding the failure.</p>

IBMSNAP_REG_SYNC table (non-DB2 relational)

The IBMSNAP_REG_SYNC table uses an update trigger to initiate an update of the SYNCHPOINT value for all the rows in the IBMSNAP_REGISTER table when the Apply program is preparing to fetch data from a non-DB2 relational data source.

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.

Table 77 provides a brief description of the columns in the IBMSNAP_REG_SYNC table.

Table 77. IBMSNAP_REG_SYNC table columns

Column name	Description
TRIGGER_ME	<p>Data type: CHAR(1); Nullable: No</p> <p>A flag of Y that indicates whether a trigger was initiated to update the SYNCHPOINT value for all rows in the register table.</p>

Table 77. IBMSNAP_REG_SYNCH 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 IBMSNAP_REGISTER table.

IBMSNAP_RESTART table

The IBMSNAP_RESTART table contains information that enables the Capture program to restart from the earliest required log or journal record. This table replaces the IBMSNAP_WARM_START table from SQL 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.

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.

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 sections show operating system-specific layouts of the IBMSNAP_RESTART table.

Linux, UNIX, Windows, z/OS

Table 78. Columns in the IBMSNAP_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.

Table 78. Columns in the IBMSNAP_RESTART table for Linux, UNIX, Windows, and z/OS (continued)

Column name	Description
CURR_COMMIT_TIME	Data type: TIMESTAMP; Nullable: No The local current timestamp when this table was updated by the Capture program.
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.

iSeries

For iSeries, the IBMSNAP_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 79. Columns in the IBMSNAP_RESTART table for iSeries

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.

Table 79. Columns in the IBMSNAP_RESTART table for iSeries (continued)

Column name	Description
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.
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.

IBMSNAP_SEQTABLE table (Informix)

The IBMSNAP_SEQTABLE table contains a sequence of unique numbers that SQL replication uses as the equivalent of log sequence numbers for Informix tables. These unique identifiers are used in the IBMSNAP_REGISTER table in place of synchpoint values so that the Capture program, Apply program, and Replication Alert Monitor can communicate the point that they left off during their last cycle.

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.

Table 80 provides a brief description of the column in the IBMSNAP_SEQTABLE table.

Table 80. Column in the IBMSNAP_SEQTABLE table

Column name	Description
SEQ	Data type: INTEGER; Nullable: No A unique number used as the log or journal identifiers (synchpoints) for Informix tables.

IBMSNAP_SIGNAL table

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.

Server: Capture control server

Default schema: ASN

Index: SIGNAL_TIME

This table contains information that you can update by using SQL.

The IBMSNAP_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 IBMSNAP_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 IBMSNAP_SIGNAL table for that insert. The values in the IBMSNAP_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 81 provides a brief description of the columns in the IBMSNAP_SIGNAL table.

Table 81. Columns in the IBMSNAP_SIGNAL table

Column name	Description
SIGNAL_TIME	<p>Data type: TIMESTAMP; Nullable: No, with default; Default: current timestamp.</p> <p>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.</p>
SIGNAL_TYPE	<p>Data type: VARCHAR(30); Nullable: No</p> <p>A flag that indicates the type of signal that was posted:</p> <p>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.</p> <p>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).</p>

Table 81. Columns in the IBMSNAP_SIGNAL table (continued)

Column name	Description
SIGNAL_SUBTYPE	<p>Data type: VARCHAR(30); Nullable: Yes</p> <p>The action that the Capture program performs when a signal from a system command (SIGNAL_TYPE = CMD) occurs.</p> <p>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.</p> <p>STOP The Capture program stops capturing changes and terminates. This command can only be issued by you, not the Apply program.</p> <p>CAPSTOP The Capture program stops capturing changes for a particular registered source, which is identified by <i>source_owner.source_table</i> in the SIGNAL_INPUT_IN column. This command can only be issued by you, not the Apply program.</p> <p>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.</p>
SIGNAL_INPUT_IN	<p>Data type: VARCHAR(500); Nullable: Yes</p> <p>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:</p> <p>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 IBMSNAP_PRUNCNTL table with the next value in the sequence.</p> <p>CMD + UPDANY The Apply qualifier that identifies the Apply program in the update-anywhere configuration.</p> <p>CMD + CAPSTOP The name of the source owner and source table that the Capture program should stop capturing changes for. (<i>source_owner.source_table</i>)</p>

Table 81. Columns in the IBMSNAP_SIGNAL table (continued)

Column name	Description
SIGNAL_STATE	<p>Data type: CHAR(1); Nullable: No</p> <p>A flag that indicates the status of the signal:</p> <p>P The signal is pending; the Capture program has not received it yet. When you post a signal, set the SIGNAL_STATE to P.</p> <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.</p> <p>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.</p>
SIGNAL_LSN	<p>Data type: CHAR(10) for bit data; Nullable: Yes</p> <p>The log sequence number of the commit record. This value is set only by the Capture program.</p>

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 IBMSNAP_SIGNAL table. 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.

IBMSNAP_UOW table

The IBMSNAP_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 IBMSNAP_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 entries in this table are deleted.

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.

iSeries:

- Because Capture for iSeries can start capturing data for a subset of the replication sources, it does not delete all the rows in the IBMSNAP_UOW table if you do a partial cold start.
- 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 UOW table is pruned by retention limits, not information from the IBMSNAP_PRUNE_SET) table.

The Capture program requires that there is one IBMSNAP_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.

The Capture program also prunes the UOW table based on information that the Apply program inserts into the IBMSNAP_PRUNE_SET table.

Table 82 provides a brief description of the columns in the IBMSNAP_UOW table.

Table 82. Columns in the IBMSNAP_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 for z/OS, this column is the correlation ID. For DB2 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 for z/OS, this column is the primary authorization ID. For DB2 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 82. Columns in the IBMSNAP_UOW table (continued)

Column name	Description
IBMSNAP_REJ_CODE	<p>Data type: CHAR(1); Nullable: No, with default; Default: 0.</p> <p>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.</p> <p>0 No known conflicts occurred in the transaction.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>
IBMSNAP_APPLY_QUAL	<p>Data type: CHAR(18); Nullable: No, with default; Default: current user name.</p> <p>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.</p>

Tables 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 ASNCLP command-line program or Replication Center. For iSeries, these control tables are created automatically for you when install DataPropagator for iSeries.

Table 83 describes the control tables at the Apply server.

Table 83. Control tables at the Apply server

Table name	Description
"ASN.IBMSNAP_APPENQ table" on page 413	Used to ensure that only one Apply program is running per Apply qualifier.
ASN.IBMSNAP_APPLY_JOB table (iSeries)	Used to ensure that there is a unique Apply qualifier for each instance of the Apply program running at an Apply control server.
"ASN.IBMSNAP_APPLYTRACE table" on page 417	Contains important messages from the Apply program.
"ASN.IBMSNAP_APPLYTRAIL table" on page 418	Contains audit-trail information about the Apply program.

Table 83. Control tables at the Apply server (continued)

Table name	Description
"ASN.IBMSNAP_APPPARAMS table" on page 414	Contains parameters that you can modify to control the operations of the Apply program.
"ASN.IBMSNAP_SUBS_COLS table" on page 423	Maps columns in the target table or view to the corresponding columns in the source table or view.
"ASN.IBMSNAP_SUBS_EVENT table" on page 424	Contains events that you define to control when the Apply program processes a subscription set.
"ASN.IBMSNAP_SUBS_MEMBR table" on page 425	Identifies a source and target table pair and specifies processing information for that pair.
"ASN.IBMSNAP_SUBS_SET table" on page 429	Contains processing information for each set of subscription-set members that the Apply program processes as a group.
"ASN.IBMSNAP_SUBS_STMTS table" on page 434	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.

ASN.IBMSNAP_APPENQ table

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 iSeries.

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.

Table 84 provides a brief description of the column in the IBMSNAP_APPENQ table.

Table 84. Column in the IBMSNAP_APPENQ 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 (iSeries)

The IBMSNAP_APPLY_JOB table, which is iSeries-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.

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.

Table 85 provides a brief description of the columns in the IBMSNAP_APPLY_JOB table.

Table 85. Columns in the IBMSNAP_APPLY_JOB table

Column name	Description
APPLY_QUAL	<p>Data type: CHAR(18); Nullable: No</p> <p>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.</p>
CONTROL_SERVER	<p>Data type: CHAR(18); Nullable: No</p> <p>The name of the database where the Apply control tables and view are defined.</p>
JOB_NAME	<p>Data type: CHAR(10); Nullable: No</p> <p>The fully qualified name of the job that wrote this trace entry:</p> <p>Position 1-10 APPLY_QUAL</p> <p>Position 11-20 The ID of the user who started the Apply program</p> <p>Position 21-26 The job number</p>
USER_NAME	<p>Data type: CHAR(10); Nullable: No</p> <p>The name of the user who started a new instance of the Apply program.</p>
JOB_NUMBER	<p>Data type: CHAR(6); Nullable: No</p> <p>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.</p>

ASN.IBMSNAP_APPPARMS table

The IBMSNAP_APPPARMS 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.

Server: Apply control server

Index: APPLY_QUAL

This table contains information that you can update by using SQL.

Table 86 on page 415 provides a brief description of the columns in the IBMSNAP_APPPARMS table.

Table 86. Columns in the IBMSNAP_APPPARMS table

Column name	Description
APPLY_QUAL	<p>Data type: CHAR(18); Nullable: No</p> <p>The Apply qualifier matches the parameters to the Apply program to which these parameters apply.</p>
APPLY_PATH	<p>Data type: VARCHAR(1040); Nullable: Yes</p> <p>The location of the work files used by the Apply program. The default is the directory where the program was started.</p>
COPYONCE	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: N.</p> <p>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.</p> <p>Y The Apply program executes one copy cycle for each eligible subscription set.</p> <p>N The Apply program does not execute one copy cycle for each eligible subscription set.</p>
DELAY	<p>Data type: INT; Nullable: Yes, with default; Default: 6.</p> <p>The delay time (in seconds) at the end of each Apply cycle when continuous replication is used. This parameter is ignored if copyonce is specified.</p>
ERRWAIT	<p>Data type: INT; Nullable: Yes, with default; Default: 300.</p> <p>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.</p>
INAMSG	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: Y.</p> <p>A flag that indicates whether the Apply program issues a message when it is inactive.</p> <p>Y The Apply program issues a message when inactive.</p> <p>N The Apply program does not issue a message when inactive.</p>
LOADXIT	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: N.</p> <p>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.</p> <p>Y The Apply program invokes ASNLOAD.</p> <p>N The Apply program does not invoke ASNLOAD.</p>
LOGREUSE	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: N.</p> <p>A flag that indicates whether the Apply program overwrites the Apply log file or appends to it.</p> <p>Y The Apply program reuses the log file by first deleting it and then re-creating it when the Apply program is restarted.</p> <p>N The Apply program appends new information to the Apply log file.</p>

Table 86. Columns in the IBMSNAP_APPPARMS table (continued)

Column name	Description
LOGSTDOUT	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: N.</p> <p>A flag that indicates where the Apply program directs the log file messages:</p> <p>Y The Apply program directs log file messages to both the standard out (STDOUT) and the log file.</p> <p>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.</p>
NOTIFY	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: N.</p> <p>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.</p> <p>Y The Apply program invokes ASNDONE.</p> <p>N The Apply program does not invoke ASNDONE.</p>
OPT4ONE	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: N.</p> <p>A flag that indicates whether the performance of the Apply program is optimized if only one subscription set is defined for the Apply program.</p> <p>Y The performance of the Apply program is optimized for one subscription set.</p> <p>N The performance of the Apply program is not optimized for one subscription set.</p> <p>This parameter is ignored if copyonce is specified.</p>
SLEEP	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: Y.</p> <p>A flag that indicates how the Apply program is to proceed if no new subscription sets are eligible for processing:</p> <p>Y The Apply program goes to sleep.</p> <p>N The Apply program stops.</p> <p>This parameter is ignored if copyonce is specified.</p>
SQLERRCONTINUE	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: N.</p> <p>A flag that indicates whether the Apply program continues processing after it checks the SQLSTATE file for errors.</p> <p>Y The Apply program checks the SQLSTATE file for any SQL errors during processing. If an error is found, Apply stops processing.</p> <p>N The Apply program does not check the SQLSTATE file and continues processing.</p>

Table 86. Columns in the IBMSNAP_APPPARMS table (continued)

Column name	Description
SPILLFILE	<p>Data type: VARCHAR(10); Nullable: Yes, with default.</p> <p>A flag that indicates where the fetched answer set is stored.</p> <p>UNIX and Windows, valid values are:</p> <p>disk (default) A disk file.</p> <p>z/OS, valid values are:</p> <p>mem (default) A memory file.</p> <p>disk A disk file.</p>
TERM	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: Y.</p> <p>A flag that indicates whether the Apply program terminates when DB2 terminates:</p> <p>Y The Apply program terminates when DB2 terminates.</p> <p>N The Apply program stays active and waits for DB2 to be restarted. This parameter is ignored if copyonce is specified.</p>
TRLREUSE	<p>Data type: CHAR(1); Nullable: Yes, with default; Default: N.</p> <p>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:</p> <p>Y The Apply program invokes ASNLOAD.</p> <p>y The Apply program does not invoke ASNLOAD.</p>

ASN.IBMSNAP_APPLYTRACE table

The IBMSNAP_APPLYTRACE table contains messages from the Apply program. The Apply program does not automatically prune this table, but you can automate pruning by adding an SQL statement that runs after one of the subscription sets.

Server: Apply control server

Index: APPLY_QUAL, TRACE_TIME

Table 87 provides a brief description of the column in the IBMSNAP_APPLYTRACE table.

Table 87. Columns in the IBMSNAP_APPLYTRACE table

Column name	Description
APPLY_QUAL	<p>Data type: CHAR(18); Nullable: No</p> <p>Uniquely identifies which Apply program inserted the message.</p>
TRACE_TIME	<p>Data type: TIMESTAMP; Nullable: No</p> <p>The time at the Apply control server when the row was inserted into this table.</p>

Table 87. Columns in the IBMSNAP_APPLYTRACE table (continued)

Column name	Description
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 table

The IBMSNAP_APPLYTRAIL 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.

Server: Apply control server

Index: LASTRUN, APPLY_QUAL

Table 88 provides a brief description of the columns in the IBMSNAP_APPLYTRAIL table.

Table 88. Columns in the IBMSNAP_APPLYTRAIL 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 IBMSNAP_SUBS_SET table after the most recent Apply cycle.				
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. <table border="0"> <tr> <td style="vertical-align: top; padding-right: 10px;">F</td> <td>(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.</td> </tr> <tr> <td style="vertical-align: top; padding-right: 10px;">S</td> <td>(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.</td> </tr> </table>	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.
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 88. Columns in the IBMSNAP_APPLYTRAIL table (continued)

Column name	Description
ASNLOAD	<p>Data type: CHAR(1); Nullable: Yes</p> <p>The value used to start the Apply program:</p> <p>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.</p> <p>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.</p> <p>NULL Indicates that an Apply program error occurred before the Apply program could determine whether the ASNLOAD exit routine should be called.</p>
FULL_REFRESH	<p>Data type: CHAR(1); Nullable: Yes</p> <p>A flag that indicates whether a full refresh occurred:</p> <p>Y Indicates that a full refresh was done for a subscription set.</p> <p>N Indicates that a full refresh was not done for a subscription set.</p> <p>NULL Indicates that an error occurred before the Apply program could determine whether or not a full refresh was needed.</p>
EFFECTIVE_MEMBERS	<p>Data type: INT; Nullable: Yes</p> <p>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.</p>
SET_INSERTED	<p>Data type: INT; Nullable: No</p> <p>The total number of rows inserted into subscription-set members during the subscription cycle.</p>
SET_DELETED	<p>Data type: INT; Nullable: No</p> <p>The total number of rows deleted from subscription-set members during the subscription cycle.</p>
SET_UPDATED	<p>Data type: INT; Nullable: No</p> <p>The total number of rows updated in subscription-set members during the subscription cycle.</p>
SET_REWORKED	<p>Data type: INT; Nullable: No</p> <p>The total number of rows that the Apply program reworked during the last cycle. The Apply program reworks changes under the following conditions:</p> <ul style="list-style-type: none"> • 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	<p>Data type: INT; Nullable: No</p> <p>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.</p>

Table 88. Columns in the IBMSNAP_APPLYTRAIL table (continued)

Column name	Description
STATUS	<p>Data type: SMALLINT; Nullable: No</p> <p>A value that represents the work status for the Apply program after a given cycle:</p> <p>-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 <i>not</i> one of the acceptable errors you indicated in the input file for SQLERRCONTINUE (<i>apply_qualifier.SQS</i>).</p> <p>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 <i>apply_qualifier.SQS</i>) and did not reject any rows.</p> <p>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.</p> <p>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 <i>apply_qualifier.SQS</i>) and rejected some of the rows. See the <i>apply_qualifier.ERR</i> file for details about the rows that failed.</p> <p>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).</p> <p>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 <i>apply_qualifier.SQS</i>) and rejected some of the rows. See the <i>apply_qualifier.ERR</i> file for details about the rows that failed.</p>
LASTRUN	<p>Data type: TIMESTAMP; Nullable: No</p> <p>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.</p>
LASTSUCCESS	<p>Data type: TIMESTAMP; Nullable: Yes</p> <p>The Apply control server timestamp for the beginning of the last successful processing of a subscription set.</p>
SYNCHPOINT	<p>Data type: CHAR(10) for bit data; Nullable: Yes</p> <p>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.</p>

Table 88. Columns in the IBMSNAP_APPLYTRAIL table (continued)

Column name	Description
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 database name where the source tables and views are defined.
SOURCE_ALIAS	Data type: CHAR(8); Nullable: Yes The DB2 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 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.
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.

Table 88. Columns in the IBMSNAP_APPLYTRAIL 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 target servers.
JRN_LIB	Data type: CHAR(10); Nullable: Yes This column, which applies only to iSeries 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 iSeries 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 IBMSNAP_SUBS_SET table.
OPTION_FLAGS	Data type: CHAR(4); Nullable: No Reserved for future options of SQL 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.
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.

Table 88. Columns in the IBMSNAP_APPLYTRAIL table (continued)

Column name	Description
APPERRM	Data type: VARCHAR(760); Nullable: Yes The Apply error message ID and text for a failed execution.

ASN.IBMSNAP_SUBS_COLS table

The IBMSNAP_SUBS_COLS 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.

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.

Table 89 provides a brief description of the columns in the IBMSNAP_SUBS_COLS table.

Table 89. Columns in the IBMSNAP_SUBS_COLS 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.				
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. <table border="0"> <tr> <td style="vertical-align: top;">F</td> <td>(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.</td> </tr> <tr> <td style="vertical-align: top;">S</td> <td>(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.</td> </tr> </table>	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.
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.				

Table 89. Columns in the IBMSNAP_SUBS_COLS table (continued)

Column name	Description
COL_TYPE	<p>Data type: CHAR(1); Nullable: No</p> <p>A flag that indicates the type of column:</p> <p>A An after-image column.</p> <p>B A before-image column.</p> <p>C A computed column or an SQL expression using scalar functions.</p> <p>F A computed column using column functions.</p> <p>L A LOB indicator value.</p> <p>P A before-image predicate column.</p> <p>R A relative record number column, provided by the system and used as a primary key column. Used only by DB2 DataPropagator for iSeries.</p>
TARGET_NAME	<p>Data type: VARCHAR(30); Nullable: No</p> <p>The name of the target table or view column. It does not need to match the source column name.</p> <p>Internal-CCD column names cannot be renamed. They must match the source-table column names.</p>
IS_KEY	<p>Data type: CHAR(1); Nullable: No</p> <p>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:</p> <p>Y The column is all or part of the target key.</p> <p>N The column is not part of the target key.</p>
COLNO	<p>Data type: SMALLINT; Nullable: No</p> <p>The numeric location of the column in the original source, to be preserved relative to other user columns in displays and subscriptions.</p>
EXPRESSION	<p>Data type: VARCHAR(254); Nullable: No</p> <p>The source column name or an SQL expression used to create the target column contents.</p>

ASN.IBMSNAP_SUBS_EVENT table

The IBMSNAP_SUBS_EVENT 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.

Server: Apply control server

Index: EVENT_NAME, EVENT_TIME

This table contains information that you can update using SQL.

You insert a row into this table when you create a new event to start an Apply program.

Table 90 on page 425 provides a brief description of the columns in the IBMSNAP_SUBS_EVENT table.

Table 90. Columns in the IBMSNAP_SUBS_EVENT 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 table

The IBMSNAP_SUBS_MEMBR 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.

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.

Table 91 provides a brief description of the columns in the IBMSNAP_SUBS_MEMBR table.

Table 91. Columns in the IBMSNAP_SUBS_MEMBR 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.

Table 91. Columns in the IBMSNAP_SUBS_MEMBR table (continued)

Column name	Description
WHOS_ON_FIRST	<p>Data type: CHAR(1); Nullable: No</p> <p>The following values are used to control the order of processing in update-anywhere replication scenarios.</p> <p>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.</p> <p>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.</p>
SOURCE_OWNER	<p>Data type: VARCHAR(30); VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; Nullable: No</p> <p>The high-level qualifier for the source table or view for this member.</p>
SOURCE_TABLE	<p>Data type: VARCHAR(128); VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; Nullable: No</p> <p>The name of the source table or view for this member.</p>
SOURCE_VIEW_QUAL	<p>Data type: SMALLINT; Nullable: No</p> <p>Supports the view of physical tables by matching the similar column in the IBMSNAP_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.</p>
TARGET_OWNER	<p>Data type: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; Nullable: No</p> <p>The high-level qualifier for the target table or view for this member.</p>
TARGET_TABLE	<p>Data type: VARCHAR(128), VARCHAR(18) for DB2 UDB for z/OS Version 8 compatibility mode subsystems or earlier; Nullable: No</p> <p>The name of the target table or view for this member.</p>
TARGET_CONDENSED	<p>Data type: CHAR(1); Nullable: No</p> <p>A flag that indicates:</p> <p>Y For any given primary key value, the target table shows only one row.</p> <p>N All changes must remain to retain a complete update history.</p> <p>A The target table is a base aggregate or change aggregate tables.</p>
TARGET_COMPLETE	<p>Data type: CHAR(1); Nullable: No</p> <p>A flag that indicates:</p> <p>Y The target table contains a row for every primary key value of interest.</p> <p>N The target table contains some subset of rows of primary key values.</p>

Table 91. Columns in the IBMSNAP_SUBS_MEMBR table (continued)

Column name	Description
TARGET_STRUCTURE	<p>Data type: SMALLINT; Nullable: No</p> <p>The structure of the target table:</p> <ul style="list-style-type: none"> 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	<p>Data type: VARCHAR(1024); Nullable: Yes</p> <p>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.</p> <p>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.</p>
MEMBER_STATE	<p>Data type: CHAR(1); Nullable: Yes</p> <p>A flag that indicates what state the member is in:</p> <ul style="list-style-type: none"> 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.

Table 91. Columns in the IBMSNAP_SUBS_MEMBR table (continued)

Column name	Description
TARGET_KEY_CHG	<p>Data type: CHAR(1); Nullable: No</p> <p>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:</p> <p>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.</p> <p>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.</p>
UOW_CD_PREDICATES	<p>Data type: VARCHAR(1024); Nullable: Yes</p> <p>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.</p>
JOIN_UOW_CD	<p>Data type: CHAR(1); Nullable: Yes</p> <p>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.</p> <p>Y The Apply program uses a join of the CD and UOW tables when processing the member.</p> <p>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.</p> <p>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.</p>

Table 91. Columns in the IBMSNAP_SUBS_MEMBR table (continued)

Column name	Description
LOADX_TYPE	<p>Data type: SMALLINT; Nullable: Yes</p> <p>The type of load for this member. The value in this column is used to override the defaults.</p> <p>NULL</p> <p>z/OS: The LOAD FROM CURSOR function (available with the DB2 Utilities Suite) is used for this member.</p> <p>Linux, UNIX and Windows: The ASNLOAD exit determines the most appropriate utility for this member (option 3, 4, or 5).</p> <p>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.</p> <p>2 A user-defined or user-modified ASNLOAD exit code is used.</p> <p>3 The LOAD FROM CURSOR function (available with the DB2 Utilities Suite) is used for this member.</p> <p>4 UNIX and Windows only: EXPORT and LOAD is used for this member.</p> <p>5 UNIX and Windows only: EXPORT and IMPORT is used for this member.</p> <p>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.</p>
LOADX_SRC_N_OWNER	<p>Data type: VARCHAR(30); Nullable: Yes</p> <p>The user-created nickname owner. This value is required when all of the following conditions exist:</p> <ul style="list-style-type: none"> • The LOAD FROM CURSOR function is used for this member (LOADX_TYPE is 3) • The target server is UNIX or Windows • The source is not a nickname
LOADX_SRC_N_TABLE	<p>Data type: VARCHAR(128); Nullable: Yes</p> <p>The user-created nickname table. This value is required when all of the following conditions exist:</p> <ul style="list-style-type: none"> • The LOAD FROM CURSOR 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 table

The IBMSNAP_SUBS_SET 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.

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.

Table 92 provides a brief description of the columns in the IBMSNAP_SUBS_SET table.

Table 92. Columns in the IBMSNAP_SUBS_SET 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.
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 alias corresponding to the Capture control server that is named in the SOURCE_SERVER column.

Table 92. Columns in the IBMSNAP_SUBS_SET table (continued)

Column name	Description
TARGET_SERVER	<p>Data type: CHAR(18); Nullable: No</p> <p>The database name of the server where target tables or views are stored.</p>
TARGET_ALIAS	<p>Data type: CHAR(8); Nullable: Yes</p> <p>The DB2 alias corresponding to the target server named in the TARGET_SERVER column.</p>
STATUS	<p>Data type: SMALLINT; Nullable: No</p> <p>A value that represents the work status for the Apply program after a given cycle:</p> <p>-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 <i>not</i> one of the acceptable errors you indicated in the input file for SQLERRCONTINUE (<i>apply_qualifier.SQS</i>).</p> <p>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 <i>apply_qualifier.SQS</i>) and did not reject any rows.</p> <p>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_SYNC_MINUTES control column.</p> <p>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 <i>apply_qualifier.SQS</i>) and rejected some of the rows. See the <i>apply_qualifier.ERR</i> file for details about the rows that failed.</p> <p>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).</p> <p>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_SYNC_MINUTES control column. However, it encountered some SQL errors that you indicated for the SQLERRCONTINUE startup parameter (in <i>apply_qualifier.SQS</i>) and rejected some of the rows. See the <i>apply_qualifier.ERR</i> file for details about the rows that failed.</p>
LASTRUN	<p>Data type: TIMESTAMP; Nullable: No</p> <p>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.</p>

Table 92. Columns in the IBMSNAP_SUBS_SET table (continued)

Column name	Description
REFRESH_TYPE	<p>Data type: CHAR(1); Nullable: No</p> <p>The type of scheduling that is used to prompt the Apply program to process this subscription set:</p> <p>R The Apply program uses time-based scheduling. It uses the value in SLEEP_MINUTES to determine when to start processing the subscription set.</p> <p>E The Apply program uses event-based scheduling. It checks the time value in the 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.</p> <p>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.</p>
SLEEP_MINUTES	<p>Data type: INT; Nullable: Yes</p> <p>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.</p>
EVENT_NAME	<p>Data type: CHAR(18); Nullable: Yes</p> <p>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.</p>
LASTSUCCESS	<p>Data type: TIMESTAMP; Nullable: Yes</p> <p>The Apply control server timestamp for the beginning of the last successful processing of a subscription set.</p>
SYNCHPOINT	<p>Data type: CHAR(10) for bit data; Nullable: Yes</p> <p>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.</p>
SYNCHTIME	<p>Data type: TIMESTAMP; Nullable: Yes</p> <p>The Apply program uses this column to record its progress, indicating that it has processed data up to this timestamp for the subscription set.</p>
CAPTURE_SCHEMA	<p>Data type: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; Nullable: No</p> <p>The schema name of the Capture control tables that process the source for this subscription set.</p>
TGT_CAPTURE_SCHEMA	<p>Data type: VARCHAR(30), VARCHAR(128) for DB2 UDB for z/OS Version 8 new-function mode subsystems; Nullable: Yes</p> <p>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.</p>

Table 92. Columns in the IBMSNAP_SUBS_SET table (continued)

Column name	Description
FEDERATED_SRC_SRVR	<p>Data type: VARCHAR(18); Nullable: Yes</p> <p>The name of the federated remote server that is the source for the subscription set, which applies only to non-DB2 relational sources.</p>
FEDERATED_TGT_SRVR	<p>Data type: VARCHAR(18); Nullable: Yes</p> <p>The name of the federated remote server that is the target for the subscription set, which applies only to non-DB2 relational targets.</p>
JRN_LIB	<p>Data type: CHAR(10); Nullable: Yes</p> <p>This column, which applies only to iSeries Capture servers, is the library name of the journal that the source table uses.</p>
JRN_NAME	<p>Data type: CHAR(10); Nullable: Yes</p> <p>This column, which applies only to iSeries 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.</p>
OPTION_FLAGS	<p>Data type: CHAR(4); Nullable: No</p> <p>Reserved for future options of SQL replication. Currently this column contains the default value of NNNN.</p>
COMMIT_COUNT	<p>Data type: SMALLINT; Nullable: Yes</p> <p>A flag that indicates the type of processing that the Apply program performs for a subscription set:</p> <p>NULL This is the default setting for a read-only subscription set. The Apply program will process fetched answer sets for the <i>n</i> 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.</p> <p><i>Integer not NULL</i></p> <p>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.</p>
MAX_SYNCH_MINUTES	<p>Data type: SMALLINT; Nullable: Yes</p> <p>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.</p>

Table 92. Columns in the IBMSNAP_SUBS_SET table (continued)

Column name	Description
AUX_STMTS	<p>Data type: SMALLINT; Nullable: No</p> <p>The number of SQL statements that you define in the IBMSNAP_SUBS_STMTS table that can run before or after the Apply program processes a subscription set.</p>
ARCH_LEVEL	<p>Data type: CHAR(4); Nullable: No</p> <p>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.</p> <p>0801 Version 8 or later SQL replication</p> <p>0803 Version 8 SQL replication with enhanced support for Oracle sources</p> <p>0805 Version 8 SQL replication with support for DB2 for z/OS new-function mode</p>

ASN.IBMSNAP_SUBS_STMTS table

The IBMSNAP_SUBS_STMTS 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.

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 AUX_STMTS column of the IBMSNAP_SUBS_SET table. If AUX_STMTS is zero for a subscription set, the corresponding entries in the IBMSNAP_SUBS_STMTS table are ignored by the Apply program.

Table 93 provides a brief description of the columns in the IBMSNAP_SUBS_STMTS table.

Table 93. Columns in the IBMSNAP_SUBS_STMTS table

Column name	Description
APPLY_QUAL	<p>Data type: CHAR(18); Nullable: No</p> <p>Uniquely identifies which Apply program processes the SQL statement or stored procedure.</p>
SET_NAME	<p>Data type: CHAR(18); Nullable: No</p> <p>The name of the subscription set that the SQL statement or stored procedure is associated with.</p>

Table 93. Columns in the IBMSNAP_SUBS_STMTS table (continued)

Column name	Description
WHOS_ON_FIRST	<p>Data type: CHAR(1); Nullable: No</p> <p>The following values are used to control the order of processing in update-anywhere replication scenarios.</p> <p>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.</p> <p>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.</p>
BEFORE_OR_AFTER	<p>Data type: CHAR(1); Nullable: No</p> <p>A value that indicates when and where the statement is issued:</p> <p>A The statement is executed at the target server after all of the answer-set rows are applied.</p> <p>B The statement is executed at the target server before any of the answer-set rows are applied.</p> <p>S The statement is executed at the Capture control server before opening the answer-set cursors.</p> <p>G Reserved for use by SQL replication.</p> <p>X Reserved for use by SQL replication.</p>
STMT_NUMBER	<p>Data type: SMALLINT; Nullable: No</p> <p>Defines the relative order of execution within the scope of the BEFORE_OR_AFTER column value.</p>
EI_OR_CALL	<p>Data type: CHAR(1); Nullable: No</p> <p>A value that indicates:</p> <p>E The SQL statement should be run as an EXEC SQL EXECUTE IMMEDIATE.</p> <p>C The SQL statement contains a stored procedure name to run as an EXEC SQL CALL.</p>
SQL_STMT	<p>Data type: VARCHAR(1024); Nullable: Yes</p> <p>One of the following values:</p> <p>Statement The SQL statement should run as an EXEC SQL EXECUTE IMMEDIATE statement if EI_OR_CALL is E.</p> <p>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.</p>

Table 93. Columns in the IBMSNAP_SUBS_STMTS 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.

List of 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. Replication on iSeries does not have Monitor control tables.

Table 94 describes the control tables at the Monitor control server.

Table 94. Control tables at the Monitor control server

Table name	Description
"IBMSNAP_ALERTS table" on page 437	Contains a record of all the alerts issued by the Replication Alert Monitor.
"IBMSNAP_CONDITIONS table" on page 438	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 table" on page 443	Contains the individual contacts that make up the contact groups.
"IBMSNAP_CONTACTS table" on page 444	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 table" on page 445	Contains the name and description of each contact group.
"IBMSNAP_MONENQ table" on page 445	Used to ensure that only one Replication Alert Monitor program is running per Monitor qualifier.
"IBMSNAP_MONPARMS table" on page 445	Contains parameters that you can modify to control the operations of the Monitor program.
"IBMSNAP_MONSERVICES table" on page 447	Contains the latest time that a server was monitored by a Replication Alert Monitor program (identified by a Monitor qualifier).
"IBMSNAP_MONTRACE table" on page 448	Contains messages from the Monitor program.
"IBMSNAP_MONTRAIL table" on page 449	Contains information about each monitor cycle.
"IBMSNAP_SUSPENDS table" on page 451	Contains information about temporary suspensions of the monitor program
"IBMSNAP_TEMPLATES table" on page 451	Contains information about how often and how long the monitor program is suspended.

IBMSNAP_ALERTS table

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.

Server: Monitor control server

Non-unique index: MONITOR_QUAL, COMPONENT, SERVER_NAME, SCHEMA_OR_QUAL, SET_NAME, CONDITION_NAME, ALERT_CODE

Table 95 provides a brief description of the columns in the IBMSNAP_ALERTS table.

Table 95. Columns in the IBMSNAP_ALERTS table

Column name	Description
MONITOR_QUAL	<p>Data type: CHAR(18); Nullable: No.</p> <p>The Monitor qualifier that identifies which Replication Alert Monitor program issued the alert.</p>
COMPONENT	<p>Data type: CHAR(1); Nullable: No.</p> <p>The replication component that is being monitored:</p> <p>C Capture program</p> <p>A Apply program</p> <p>S Q Capture program</p> <p>R Q Apply program</p>
SERVER_NAME	<p>Data type: CHAR(18); Nullable: No.</p> <p>The name of the Capture control server, Apply control server, Q Capture server, or Q Apply server where the alert condition occurred.</p>
SERVER_ALIAS	<p>Data type: CHAR(8); Nullable: Yes.</p> <p>The DB2 alias of the Capture control server, Apply control server, Q Capture server, or Q Apply server where the alert condition occurred.</p>
SCHEMA_OR_QUAL	<p>Data type: VARCHAR(128); Nullable: No.</p> <p>The Capture schema, Apply schema, Q Capture schema, or Q Apply schema that is being monitored.</p>
SET_NAME	<p>Data type: CHAR(18); Nullable: No, with default; Default: Current subscription set.</p> <p>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.</p> <p>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.</p>
CONDITION_NAME	<p>Data type: CHAR(18); Nullable: No.</p> <p>The condition code that was tested when the alert was triggered.</p>

Table 95. Columns in the IBMSNAP_ALERTS table (continued)

Column name	Description
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

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.

Server: Monitor control server

Non-unique index: MONITOR_QUAL, COMPONENT, SERVER_NAME, SCHEMA_OR_QUAL, SET_NAME, CONDITION_NAME

Table 96 provides a brief description of the columns in the IBMSNAP_CONDITIONS table.

Table 96. 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.

Table 96. Columns in the IBMSNAP_CONDITIONS table (continued)

Column name	Description
COMPONENT	<p>Data type: CHAR(1); Nullable: No.</p> <p>The replication component that is being monitored:</p> <p>C Capture program</p> <p>A Apply program</p> <p>S Q Capture program</p> <p>R Q Apply program</p>
SCHEMA_OR_QUAL	<p>Data type: VARCHAR(128); Nullable: No.</p> <p>The Capture schema, Apply schema, Q Capture schema, or Q Apply schema that is being monitored.</p>
SET_NAME	<p>Data type: CHAR(18); Nullable: No; Default: Current subscription set.</p> <p>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.</p> <p>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.</p>
MONITOR_QUAL	<p>Data type: CHAR(18); Nullable: No.</p> <p>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.</p>
SERVER_ALIAS	<p>Data type: CHAR(8); Nullable: Yes.</p> <p>The DB2 alias of the Capture control server, Apply control server, Q Capture server, or Q Apply server where this condition is being monitored.</p>
ENABLED	<p>Data type: CHAR(1); Nullable: No.</p> <p>A flag that indicates whether the Replication Alert Monitor will process this condition during the next monitoring cycle:</p> <p>Y The Replication Alert Monitor will process this definition during the next cycle.</p> <p>N The Replication Alert Monitor will ignore this definition during the next cycle.</p>

Table 96. Columns in the IBMSNAP_CONDITIONS table (continued)

Column name	Description
CONDITION_NAME	<p>Data type: CHAR(18); Nullable: No.</p> <p>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.</p> <p>CAPTURE_STATUS The status of the Capture program.</p> <p>CAPTURE_ERRORS Whether the Capture program posted any error messages.</p> <p>CAPTURE_WARNINGS Whether the Capture program posted any warning messages.</p> <p>CAPTURE_LASTCOMMIT The last time the Capture program committed data during the last monitor cycle.</p> <p>CAPTURE_CLATENCY The Capture program's current latency.</p> <p>CAPTURE_HLATENCY Whether the Capture program's latency is greater than a certain number of seconds.</p> <p>CAPTURE_MEMORY The amount of memory (in megabytes) that the Capture program is using.</p>
CONDITION_NAME (Continued)	<p>APPLY_STATUS The status of the Apply program.</p> <p>APPLY_SUBSFALING Whether any subscription sets failed.</p> <p>APPLY_SUBSINACT Whether any subscription sets either failed or are inactive.</p> <p>APPLY_ERRORS Whether the Apply program posts any error messages.</p> <p>APPLY_WARNINGS Whether the Apply program posts any warning messages.</p> <p>APPLY_FULLREFRESH Whether a full refresh occurred.</p> <p>APPLY_REJTRANS (update anywhere) Whether the Apply program rejects transactions in any subscription set.</p> <p>APPLY_SUBSDELAY Whether the Apply program delays more than the time that you specified in the PARM_INT parameter.</p> <p>APPLY_REWORKED Whether the Apply program reworked any rows at the target table.</p> <p>APPLY_LATENCY Whether the end-to-end latency of the Apply program exceeds a threshold.</p>

Table 96. 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 threshold.
	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_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 96. Columns in the IBMSNAP_CONDITIONS table (continued)

Column name	Description
PARM_INT	<p>Data type: INT; Nullable: Yes.</p> <p>The integer parameter for the condition. The value of this column depends on the value of the CONDITION_NAME column.</p> <p>CAPTURE_LASTCOMMIT Threshold in seconds.</p> <p>CAPTURE_CLATENCY Threshold in seconds.</p> <p>CAPTURE_HLATENCY Threshold in seconds.</p> <p>CAPTURE_MEMORY Threshold in megabytes.</p> <p>APPLY_SUBSDELAY Threshold in seconds.</p> <p>APPLY_REWORKED Threshold in rows reworked.</p> <p>APPLY_LATENCY Threshold in seconds.</p> <p>QCAPTURE_LATENCY Threshold in seconds</p> <p>QCAPTURE_MEMORY Threshold in megabytes</p> <p>QCAPTURE_TRANSIZE Threshold in megabytes</p> <p>QAPPLY_EELATENCY Threshold in seconds</p> <p>QAPPLY_LATENCY Threshold in seconds</p> <p>QAPPLY_MEMORY Threshold in megabytes</p> <p>QAPPLY_SPILLQDEPTH Threshold in number of messages.</p> <p>QAPPLY_QDEPTH Threshold in number of messages.</p>

Table 96. Columns in the IBMSNAP_CONDITIONS table (continued)

Column name	Description
PARM_CHAR	<p>Data type: VARCHAR(128); Nullable: Yes.</p> <p>The character parameter for the condition. This column holds additional strings used by the condition.</p> <p>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:</p> <ul style="list-style-type: none"> • Capture server or Apply control server. z/OS: This is the DB2 subsystem name. • Remote DB2 instance name (only when the server is remote). • Remote hostname. <p>If the value is NULL or a zero length string, the Monitor program uses the following defaults:</p> <ul style="list-style-type: none"> • 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	<p>Data type: CHAR(1); Nullable: No.</p> <p>A flag that indicates whether to contact an individual or a group if this condition occurs:</p> <p>C Individual contact</p> <p>G Group of contacts</p>
CONTACT	<p>Data type: VARCHAR(127); Nullable: No.</p> <p>The name of the individual contact or the group of contacts to be notified if this condition occurs.</p>

IBMSNAP_CONTACTGRP table

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).

Server: Monitor control server

Non-unique index: GROUP_NAME, CONTACT_NAME

Table 97 provides a brief description of the columns in the IBMSNAP_CONTACTGRP table.

Table 97. Columns in the IBMSNAP_CONTACTGRP table

Column name	Description
GROUP_NAME	<p>Data type: VARCHAR(127); Nullable: No.</p> <p>The name of the contact group.</p>

Table 97. 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

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.

Server: Monitor control server

Non-unique index: CONTACT_NAME

Table 98 provides a brief description of the columns in the IBMSNAP_CONTACTS table.

Table 98. 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_GROUPS table

The IBMSNAP_GROUPS table contains the name and description of each contact group. One group per row is specified.

Server: Monitor control server

Non-unique index: GROUP_NAME

Table 99 provides a brief description of the columns in the IBMSNAP_GROUPS table.

Table 99. 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_MONENQ table

The IBMSNAP_MONENQ table is reserved for future use by replication.

Server: Monitor control server

Non-unique index: MONITOR_QUAL

Table 100 provides a brief description of the column in the IBMSNAP_MONENQ table.

Table 100. Columns in the IBMSNAP_MONENQ table

Column name	Description
MONITOR_QUAL	Data type: CHAR(18); Nullable: No. Reserved for future use by replication.

IBMSNAP_MONPARMS table

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 number of notification messages that the Monitor program will send when an alert condition is met. If you make changes to the parameters in this table, the Monitor program reads your modifications only during startup.

Server: Monitor control server

Index: MONITOR_QUAL

Default schema: ASN

This table contains information that you can update by using SQL.

Table 101 provides a brief description of the columns in the IBMSNAP_MONPARMS table.

Table 101. 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 IBMSNAP_ALERTS, IBMSNAP_MONTRACE, and IBMSNAP_MONTRAIL control 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 recreating 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.
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.

Table 101. Columns in the IBMSNAP_MONPARMS table (continued)

Column name	Description
MONITOR_PATH	<p>Data type: VARCHAR(1040); Nullable: Yes.</p> <p>The path where the output from the Monitor program is sent.</p>
RUNONCE	<p>Data type: CHAR(1); Nullable: No, with default; Default: N.</p> <p>A flag that indicates whether the Monitor program will check for the alert conditions that were selected:</p> <p>Y The Monitor program checks for any alert conditions.</p> <p>N The Monitor program does not check for any alert conditions.</p> <p>If RUNONCE is set to y, then the MONITOR_INTERVAL is ignored.</p>
TERM	<p>Data type: CHAR(1); Nullable: No, with default; Default: N.</p> <p>A flag that indicates whether the Monitor program terminates when DB2 terminates:</p> <p>Y The Monitor program terminates when DB2 terminates:</p> <p>N The Monitor program stays active and waits for DB2 to be restarted.</p>
TRACE_LIMIT	<p>Data type: INT; Nullable: No, with default; Default: 10080.</p> <p>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.</p>
ARCH_LEVEL	<p>Data type: CHAR(8); Nullable: No, with default; Default: 0901.</p> <p>The architectural level of the definition contained in the row. This column identifies the rules under which a row was created. This level is defined by IBM, and for Version 9.1 the level is 0901.</p> <p>Important: When updating the IBMSNAP_MONPARMS table, do not change the value in this column.</p>

IBMSNAP_MONSERVERS table

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.

Server: Monitor control server

Non-unique index: MONITOR_QUAL, SERVER_NAME

Table 102 provides a brief description of the columns in the IBMSNAP_MONSERVERS table.

Table 102. Columns in the IBMSNAP_MONSERVERS table

Column name	Description
MONITOR_QUAL	<p>Data type: CHAR(18); Nullable: No.</p> <p>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.</p>

Table 102. Columns in the IBMSNAP_MONSERVERS table (continued)

Column name	Description
SERVER_NAME	<p>Data type: CHAR(18); Nullable: No.</p> <p>The name of the Capture control server, Apply control server, Q Capture server, or Q Apply server that the Replication Alert Monitor was monitoring.</p>
SERVER_ALIAS	<p>Data type: CHAR(8); Nullable: Yes.</p> <p>The DB2 alias of the Capture control server, Apply control server, Q Capture server, or Q Apply server that the Replication Alert Monitor was monitoring.</p>
LAST_MONITOR_TIME	<p>Data type: TIMESTAMP; Nullable: Yes.</p> <p>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.</p>
START_MONITOR_TIME	<p>Data type: TIMESTAMP; Nullable: Yes.</p> <p>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 an upper bound value to fetch alert messages from the control tables.</p>
END_MONITOR_TIME	<p>Data type: TIMESTAMP; Nullable: Yes.</p> <p>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.</p>
LASTRUN	<p>Data type: TIMESTAMP; Nullable: No.</p> <p>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.</p>
LASTSUCCESS	<p>Data type: TIMESTAMP; Nullable: Yes.</p> <p>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 column is stored in the IBMSNAP_MONTRAIL table).</p>
STATUS	<p>Data type: SMALLINT; Nullable: No.</p> <p>A flag that indicates the status of the monitoring cycle:</p> <ul style="list-style-type: none"> -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

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.

Server: Monitor control server

Non-unique index: MONITOR_QUAL, TRACE_TIME

Table 103 provides a brief description of the columns in the IBMSNAP_MONTRACE table.

Table 103. 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

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.

Server: Monitor control server

Non-unique index: None

Table 104 provides a brief description of the columns in the IBMSNAP_MONTRAIL table.

Table 104. 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 104. Columns in the IBMSNAP_MONTRAIL table (continued)

Column name	Description
SERVER_ALIAS	Data type: CHAR(8); Nullable: Yes The DB2 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 The time when this row was inserted into this table. Default: Current timestamp
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.

Table 104. Columns in the IBMSNAP_MONTRAIL table (continued)

Column name	Description
SUSPENSION_NAME	Data type: VARCHAR(128); Nullable: Yes The name of the suspension that is used to stop the operation of the monitor for defined periods.

IBMSNAP_SUSPENDS table

The IBMSNAP_SUSPENDS table stores information about temporary suspensions of the monitor program.

Server: Monitor control server

Default schema: ASN

Primary key: SUSPENSION_NAME

Unique index: SERVER_NAME, TEMPLATE_NAME, START

Table 105 provides a brief description of the columns in the IBMSNAP_SUSPENDS table.

Table 105. Columns in the IBMSNAP_SUSPENDS table

Column name	Description
SUSPENSION_NAME	Data type: VARCHAR(128); Nullable: No The name of the monitor suspension.
SERVER_NAME	Data type: CHAR(18); Nullable: No The name of the Q Capture server, Q Apply server, Capture control server, or Apply control server where monitoring is suspended.
SERVER_ALIAS	Data type: CHAR(18); Nullable: Yes The alias of the server where monitoring is suspended.
TEMPLATE_NAME	Data type: VARCHAR(128); Nullable: Yes The name of the monitor suspension template. If the value of this column does not exist in the IBMSNAP_TEMPLATES control table, the monitor suspends one time from the START timestamp until the STOP timestamp.
START	Data type: TIMESTAMP; Nullable: No The time to start using the template. If no template is specified, this is the start time of the suspension.
STOP	Data type: TIMESTAMP; Nullable: No The time to stop using the template. If no template is specified, this is the end time of the suspension.

IBMSNAP_TEMPLATES table

The IBMSNAP_TEMPLATES table stores information about how often and how long the monitor program is suspended. This information is called a monitor suspension template.

Server: Monitor control server

Default schema: ASN

Unique index: TEMPLATE_NAME

Table 106 provides a brief description of the columns in the IBMSNAP_TEMPLATES table.

Table 106. Columns in the IBMSNAP_TEMPLATES table

Column name	Description
TEMPLATE_NAME	Data type: VARCHAR(128); Nullable: No The name of the monitor suspension template.
START_TIME	Data type: TIME; Nullable: No The time of the day to start the suspension. Default: 00:00:00
WDAY	Data type: SMALLINT; Nullable: Yes The day of the week in which the suspension begins, starting with 0 for Sunday and continuing to 6 for Saturday. A null values means the suspension can begin on any day.
DURATION	Data type: INTEGER; Nullable: No The duration of the suspension in minutes.

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 ASNCLP command-line program or Replication Center build the target table to your specifications based on how you define the subscription-set member.

Table 107 describes the tables at the target server.

Table 107. Quick reference for target tables

Table name	Description
"Base aggregate table" on page 453	Contains data that has been aggregated from a source table.
"Change aggregate table" on page 453	Contains data that has been aggregated from a CD table.
"Consistent-change data (CCD) table" on page 454	Contains information about changes that occur at the source and contains additional columns to identify the sequential ordering of those changes.
"Point-in-time table" on page 456	A copy of the source data, with an additional column that records the specific time in the source log that the data was committed.
"Replica table" on page 457	A type of target table used for update-anywhere replication.
"User copy table" on page 457	A copy of the source table.

Base aggregate table

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.

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.

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
<i>user columns</i>	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

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.

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.

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
<i>user key columns</i>	The columns that make up the target key.
<i>user nonkey columns</i>	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.
<i>user computed columns</i>	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_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

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.

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.

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 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 IBMSNAP_REGISTER table.

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.

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.

Table 110. Columns in the CCD table (continued)

Column name	Description
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.
<i>user key columns</i>	If the CCD table is condensed, this column contains the columns that make up the target key.
<i>user nonkey columns</i>	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.
<i>user computed columns</i>	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 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 for z/OS, this column is the correlation ID. For DB2 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.

Point-in-time table

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.

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.

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
<i>user key columns</i>	The columns that make up the target key.
<i>user nonkey columns</i>	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.
<i>user computed columns</i>	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

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 other 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.

schema.replica

Server: target server

This table contains information that you can update by using SQL.

Table 112 provides a brief description of the columns in the replica table.

Table 112. Columns in the replica table

Column name	Description
<i>user key columns</i>	The columns that make up the target key, which must be the same primary key as the master table.
<i>user nonkey columns</i>	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

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.

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.

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
<i>user key columns</i>	The columns that make up the target key.
<i>user nonkey columns</i>	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.
<i>user computed columns</i>	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.

Chapter 25. UNICODE and ASCII encoding schemes for SQL replication (z/OS)

SQL replication for OS/390 and z/OS Version 7 or later supports 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 SQL replication source, target, and control tables as described in the following sections. However, your existing replication environment will work with SQL replication 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.

Rules for 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 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_xxx) 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.

It is recommended that you stop the Capture and Apply programs before you change the encoding scheme of existing tables.

Note: The *DB2 UDB for z/OS V8 SQL Reference* contains more information about CCSID.

To set encoding schemes:

1. Create new source and target tables with the proper encoding scheme. It is recommended that afterwards that you initialize Capture with a cold start and restart the Apply program.
2. If you have already created your source and target tables, change the encoding schemes of the existing target and source tables. Existing tables 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 UDB for z/OS V8 Utility Guide and Reference* for more information on the Load and Reorg utilities.
3. Use the Replication Center to create new control tables with the proper encoding scheme.
4. Use the Reorg and Load utilities to modify the encoding scheme for existing control tables and CD tables.
5. When you create new replication sources or subscription sets using the ASNCLP or Replication Center, specify the proper encoding scheme.

Chapter 26. 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.

Chapter 27. 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	Journalized 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 iSeries 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	Journalized 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.

Note:

- 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
- 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.
- 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.

Accessing information about IBM

IBM has several methods for you to learn about products and services.

You can find the latest information on the Web at www.ibm.com/software/data/integration/db2ii/support.html:

- Product documentation in PDF and online information centers
- Product downloads and fix packs
- Release notes and other support documentation
- Web resources, such as white papers and IBM Redbooks™
- Newsgroups and user groups
- Book orders

To access product documentation, go to this site:

publib.boulder.ibm.com/infocenter/db2help/topic/

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 order publications by telephone in the United States, call 1-800-879-2755.

To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide.

Contacting IBM

You can contact IBM by telephone for customer support, software services, and general information.

Customer support

To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

Software services

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

General information

To find general information in the United States, call 1-800-IBM-CALL (1-800-426-2255).

Go to www.ibm.com for a list of numbers outside of the United States.

Accessible documentation

Documentation is provided in XHTML format, which is viewable in most Web browsers.

XHTML allows you to view documentation according to the display preferences that you 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.

Providing comments on the documentation

Please send any comments that you have about this information or other documentation.

Your feedback helps IBM to provide quality information. 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/awdtools/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 information (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).

Notices and trademarks

Notices

This information was developed for products and services offered in the U.S.A.

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

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

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

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

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

IBM Corporation
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 is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which 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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM trademarks and certain non-IBM trademarks are marked at their first occurrence in this document.

See <http://www.ibm.com/legal/copytrade.shtml> for information about IBM trademarks.

The following terms are trademarks or registered trademarks of other companies:

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.

Microsoft, Windows, Windows NT®, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel®, Intel Inside® (logos), MMX and Pentium® are trademarks of Intel 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.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names might be trademarks or service marks of others.

Index

Special characters

; delimiter 235
\$TA JES2 command 220
*.APP.log file 118
*.CAP.log file 100
*.err file 121
*.sqz file 121
delimiter 235

A

abstract data types 81
accessibility 466
activating subscription sets 57
add_partition parameter
 overview 97
ADDDPRREG command 287
ADDDPRSUB command 295
ADDDPRSUBM command 310
ADDJOBSCDE command 220
administration
 authorization requirements 13
after-image columns 38
aggregate tables
 base aggregate 68, 453
 change aggregate 68, 453
alert conditions
 ASNMAIL exit routine 196
 e-mail notification 195
 for Apply program 192
 for Capture program 192
 for Q Apply program 192
 for Q Capture program 192
 list 192
 overview 192
alert_prune_limit parameter, Replication
 Alert Monitor 208
alerts
 sending to z/OS console 196
ALWINACT parameter 356
Analyzer
 for iSeries
 creating SQL packages 27
 invocation parameters 320
Analyzer report
 ANZDPR command 319
 asanalyze command 254
ANZDPR command 319
ANZDPRJRN command 32
APPLHEAPSZ configuration
 parameter 24
applications
 starting replication programs
 from 461
Apply control tables
 APPPARMS (Apply parameters)
 changing 123
 IBMSNAP_APPENQ 413
 IBMSNAP_APPLY_JOB 413
 IBMSNAP_APPLYTRACE 417
Apply control tables (*continued*)
 IBMSNAP_APPLYTRAIL 418
 IBMSNAP_APPPARMS 414
 using 122
 IBMSNAP_SUBS_COLS 423
 IBMSNAP_SUBS_EVENT 424
 IBMSNAP_SUBS_SET 429
 IBMSNAP_SUBS_STMTS 434
 SUBS_MEMBR (subscription
 members) 425
Apply parameters (APPPARMS) table
 changing 123
Apply program
 alert conditions 192
 authorization requirements 15
 changing parameter values 122
 commands 239
 asnacmd 253
 asnapply 247
 communicating with
 Capture program 221, 222
 Capture triggers 221, 223
 Replication Alert Monitor 225
 Replication Center 221
 connectivity 19
 data blocking 58
 for iSeries
 ALWINACT parameter 356
 APYQUAL parameter 354
 checking status 233
 COPYONCE parameter 357
 creating SQL packages 27
 CTLSVR parameter 355
 DELAY parameter 356
 FULLREFPGM parameter 355
 INACTMSG parameter 356
 JOB parameter 354
 OPTSNGSET parameter 357
 RTYWAIT parameter 356
 scheduling 220
 setting up 28
 starting 112, 353
 stopping 123, 328
 SUBNFYPGM parameter 356
 TRACE parameter 355
 TRLREUSE parameter 357
 USER parameter 354
 for UNIX
 apply_path parameter 115
 apply_qual parameter 116
 binding 25
 checking status 227
 configuring 25
 control_server parameter 116
 copyonce parameter 116
 default parameters 113
 delay parameter 117
 errwait parameter 117
 inamsg parameter 118
 loadxit parameter 118
 logreuse parameter 118
Apply program (*continued*)
 for UNIX (*continued*)
 logstdout parameter 119
 notify parameter 119
 opt4one parameter 119
 password file 17
 pwdfile parameter 119
 setting up 24
 sleep parameter 120
 spillfile parameter 120
 sqlerrcontinue parameter 121
 starting 111, 461
 stopping 123
 term parameter 121
 trlreuse parameter 122
 for Windows
 apply_path parameter 115
 apply_qual parameter 116
 binding 25
 checking status 227
 configuring 25
 control_server parameter 116
 copyonce parameter 116
 default parameters 113
 delay parameter 117
 errwait parameter 117
 inamsg parameter 118
 loadxit parameter 118
 logreuse parameter 118
 logstdout parameter 119
 notify parameter 119
 opt4one parameter 119
 pwdfile parameter 119
 setting up 24
 sleep parameter 120
 spillfile parameter 120
 sqlerrcontinue parameter 121
 starting 111, 461
 stopping 123
 term parameter 121
 trlreuse parameter 122
 for z/OS
 apply_path parameter 115
 apply_qual parameter 116
 checking status 227
 control_server parameter 116
 copyonce parameter 116
 db2_subsystem parameter 117
 default parameters 113
 delay parameter 117
 errwait parameter 117
 inamsg parameter 118
 loadxit parameter 118
 logreuse parameter 118
 logstdout parameter 119
 notify parameter 119
 opt4one parameter 119
 pwdfile parameter 119
 setting up 28
 sleep parameter 120

- Apply program (*continued*)
 - for z/OS (*continued*)
 - spillfile parameter 120
 - starting 111, 134
 - stopping 123
 - term parameter 121
 - trlreuse parameter 122
 - latency analysis 232
 - messages 231
 - printing 231
 - mini-cycles 58
 - operating 111
 - performance data 228
 - run-time processing statements 90
 - scheduling 219
 - setting defaults for parameters 122
 - table-mode processing 60
 - throughput analysis 232
 - transaction-mode processing 60
 - user ID 15
- Apply qualifiers
 - changing in subscription sets 157
 - monitoring status 232
 - naming rules 237
 - number of associated subscription sets 54
 - use when starting the Apply program 111, 112
- apply_path parameter 115
- apply_qual parameter 116
- APPPARMS (Apply parameters) table
 - changing 123
- APYQUAL parameter 354
- ARM (Automatic Restart Manager) 135
- ASCII tables 459
- asacmd command 253
- asanalyze command 254
- asnapply command 247
- asncap command 239
- asnccmd command 244
- ASNDONE exit routine
 - rejected transactions 46
 - using 123, 124
- asndone.smp file 123
- ASNLOAD exit routine
 - customizing behavior 129
 - description 126
 - error handling 126
 - for iSeries 131
 - for UNIX 126
 - for Windows 126
 - for z/OS 128
 - using asnload.ini file 130
 - using load from cursor function 129
- asnload.ini file 130
- ASNMAIL exit routine 196
- asnmcmd command 261
- ASNPLXFY utility 136
- asnpwd 264
- asnsct 268
- asnsdrop 271
- asnslist command 271
- asntdiff command 272
- asntdiff utility
 - overview 183
- asntrc 275
- asntrep command 282

- asntrep utility
 - usage guide 186
- asntrepair utility
 - usage guide 183
- asntrepair utility
 - overview 183
- AT command
 - Apply program 219, 220
 - Capture program 219, 220
 - Replication Alert Monitor 219, 220
- AT NetView command
 - Apply for z/OS 220
 - Capture for z/OS 220
- attributes
 - changing for registered objects 140
 - changing for subscription sets 149
- auditing
 - cold start 68
 - gap in data 68
 - source data 38
- authentication, end-user
 - for UNIX 17
 - for Windows 17
- authorization
 - for administration 13
 - for Apply program 15
 - for Capture program 14
 - for Capture triggers 16
 - for Replication Alert Monitor 198
- automatic pruning 176
- Automatic Restart Manager (ARM) 135
- autoprun parameter
 - overview 97
- autoprun parameter, Replication Alert Monitor 208
- autostop parameter 98

B

- backup database command 24
- base aggregate tables
 - definition 65
 - structure 453
 - usage 68
- batch jobs
 - memory used by 1
 - running 133
- before-image columns
 - change-aggregate tables 75
 - registering 38
 - restrictions 38
- before-image prefix 41
- binary large object (BLOB)
 - replication considerations 82
- binding
 - Apply program
 - for UNIX 25
 - for Windows 25
 - Capture program
 - for UNIX 25
 - for Windows 25
 - Replication Alert Monitor
 - for UNIX 198
 - for Windows 198
- BLOB (binary large object)
 - replication considerations 82
- blocking factor 58

C

- calculated columns 75
- CALL procedures
 - before and after run-time processing 90
 - defining for subscription set 61
- CAPCTLLIB parameter 362
- CAPPARMS (Capture parameters) table
 - changing 107
 - using 104
- CAPSTART signals 165
- CAPSTOP signals 166
- Capture
 - multiple database partitions 28
 - using multiple database partitions 23
- Capture control server
 - multiple Capture schemas 23
- Capture control tables
 - CAPPARMS (Capture parameters)
 - changing 107
 - using 104
 - CCD (consistent-change-data) 390
 - CD (change-data) 391
 - IBMSNAP_AUTHTKN 388
 - IBMSNAP_CAPENQ 382
 - IBMSNAP_CAPMON 382
 - IBMSNAP_CAPPARMS
 - structure 384
 - IBMSNAP_CAPSCHEMAS 387
 - IBMSNAP_CAPTRACE 389
 - IBMSNAP_PARTITIONINFO 392
 - IBMSNAP_PRUNCNTL 393
 - IBMSNAP_PRUNE_LOCK 395
 - IBMSNAP_PRUNE_SET 396
 - IBMSNAP_REG_EXT 396
 - IBMSNAP_REG_SYNCH 404
 - IBMSNAP_REGISTER 398
 - IBMSNAP_RESTART 405
 - IBMSNAP_SEQTABLE 407
 - IBMSNAP_SIGNAL 407
 - IBMSNAP_UOW 410
- Capture log file 100
- Capture parameters (CAPPARMS) table
 - changing 107
 - using 104
- Capture program
 - alert conditions 192
 - altering behavior while running 106
 - authorization requirements 14
 - changing parameter values 104
 - changing schemas 145
 - cold start prevention 179
 - commands 239
 - asncap 239
 - asnccmd 244
 - communicating with
 - Apply program 221, 222
 - Replication Alert Monitor 225
 - Replication Center 221
 - connectivity 19
 - for iSeries
 - authorization requirements 13
 - CAPCTLLIB parameter 362
 - changing attributes 322
 - checking status 233
 - CLNUPITV parameter 362
 - cold start parameters 361

Capture program (*continued*)
 for iSeries (*continued*)
 creating SQL packages 26
 default parameters 95
 FRCFRQ parameter 365
 JOB parameter 361
 journal entry types 463
 journals and journal receivers,
 managing 30
 JRN parameter 362
 LAG parameter 364
 MEMLMT parameter 363
 MONITV parameter 363
 MONLMT parameter 363
 operating 93
 overriding attributes of 342
 progress of 233
 reinitializing 341
 RESTART parameter 361
 RETAIN parameter 364
 scheduling 220
 setting up 28
 starting 94, 359
 stopping 108, 331
 TRCLMT parameter 363
 WAIT parameter 361
 warm start parameters 361
 for UNIX
 add_partition parameter 97
 autoprune parameter 97
 autostop parameter 98
 binding 25
 capture_path parameter 98
 capture_schema parameter 98
 capture_server parameter 99
 checking status 227
 cold start parameters 103
 commit_interval parameter 99
 configuring 24
 default parameters 95
 lag_limit parameter 100
 logreuse parameter 100
 logstdout parameter 100
 memory_limit parameter 101
 monitor_interval parameter 101
 monitor_limit parameter 101
 operating 93
 prune_interval parameter 101
 reinitializing 108
 resuming 109
 retention_limit parameter 102
 setting up 24
 sleep_interval parameter 103
 starting 93, 461
 startmode parameter 103
 stopping 108
 suspending 109
 term parameter 104
 trace_limit parameter 104
 warm start parameters 103
 for Windows
 add_partition parameter 97
 autoprune parameter 97
 autostop parameter 98
 binding 25
 capture_path parameter 98
 capture_schema parameter 98

Capture program (*continued*)
 for Windows (*continued*)
 capture_server parameter 99
 checking status 227
 cold start parameters 103
 commit_interval parameter 99
 configuring 24
 default parameters 95
 lag_limit parameter 100
 logreuse parameter 100
 logstdout parameter 100
 memory_limit parameter 101
 monitor_interval parameter 101
 monitor_limit parameter 101
 operating 93
 prune_interval parameter 101
 reinitializing 108
 resuming 109
 retention_limit parameter 102
 setting up 24
 sleep_interval parameter 103
 starting 93, 461
 startmode parameter 103
 stopping 108
 suspending 109
 term parameter 104
 trace_limit parameter 104
 warm start parameters 103
 for z/OS
 add_partition parameter 97
 autoprune parameter 97
 autostop parameter 98
 capture_path parameter 98
 capture_schema parameter 98
 capture_server parameter 99
 checking status 227
 cold start parameters 103
 commit_interval parameter 99
 default parameters 95
 lag_limit parameter 100
 logreuse parameter 100
 logstdout parameter 100
 memory_limit parameter 101
 monitor_interval parameter 101
 monitor_limit parameter 101
 operating 93
 prune_interval parameter 101
 reinitializing 108
 resuming 109
 retention_limit parameter 102
 setting up 28
 sleep_interval parameter 103
 starting 93
 startmode parameter 103
 stopping 108
 suspending 109
 term parameter 104
 trace_limit parameter 104
 warm start parameters 103
 latency analysis 230
 memory used by 1
 messages 230
 printing 230
 performance data 228
 running more than one 23
 scheduling 219
 setting defaults for parameters 104

Capture program (*continued*)
 setting environment variables 24
 signals 161
 throughput analysis 230
 user ID 14
 where to start it 99
 Capture schemas
 changing 145
 naming rules 237
 using multiple 23
 Capture signals 161
 Capture triggers
 authorization requirements 16
 communicating with
 Apply program 221, 223
 Replication Center 221
 conflicts with preexisting triggers 8
 names of 8
 planning 8
 capture_path parameter 98
 capture_schema parameter 98
 capture_server parameter 99
 catalog tables, registering 33
 CCD (consistent-change-data) tables
 adding UOW columns 68
 external
 multi-tier replication 71
 internal
 multiple targets 69
 locks on 9
 non-DB2 relational data sources
 using CCD tables 35
 nonrelational data sources
 maintaining CCD tables 52
 using CCD tables 33
 replication sources 71
 structure
 Capture control server 390
 target server 454
 usage
 history or audit 68
 multi-tier replication 71
 CD (change-data) tables
 for joins 49
 for views 49
 pruning 177
 storage requirements 5
 structure 391
 summarizing contents 68
 CD (change-data) views 49
 change aggregate tables
 definition 65
 structure 453
 usage 68
 change-capture replication
 description 37
 registration option 37
 change-data (CD) tables
 pruning 177
 storage requirements 5
 structure 391
 summarizing contents 68
 changing Capture parameters
 for iSeries 322
 character large object (CLOB)
 replication considerations 82
 CHGDPRCAPA command 322

- CHGJRN command 31
- CLNUPITV parameter 362
- CLOB (character large object)
 - replication considerations 82
- code pages
 - compatible 9
 - DB2CODEPAGE environment variable 10
 - translation 9
- cold start, Capture program
 - for iSeries 361
 - for UNIX 103
 - for Windows 103
 - for z/OS 103
 - preventing 179
- cold startmode 103
- column (vertical) subsetting
 - at the source 37
 - at the target 75
- columns
 - adding to registered source tables 140
 - after-image 38
 - available for replication 37
 - before-image 38
 - calculated 75
 - computed 91
 - defining in target table 75
 - mapping from sources to targets 76
 - registering in source table 37
 - relative record numbers on iSeries 48
 - renaming 76, 91
 - subsetting
 - at the source 37
 - at the target 75
- commands
 - asnacmd 253
 - asnapply 247
 - asncap 239
 - asnccmd 244
- comments on documentation 466
- commit_interval parameter
 - overview 99
 - tuning 1
- communication amongSQL replication
 - components 221
- compression dictionaries (z/OS) 174
- computed columns
 - CD table 68
 - creating 91
 - source table 68
- configuration parameters for DB2
 - APPLHEAPSZ 24
 - DBHEAP 24
 - LOGBUFSZ 24
 - LOGFILSIZ 24
 - LOGPRIMARY 24
 - LOGSECOND 24
 - MAXAPPLS 24
- configuring
 - Apply program
 - for UNIX 25
 - for Windows 25
 - Capture program
 - for UNIX 24
 - for Windows 24
 - connectivity 19
 - configuring (*continued*)
 - Replication Alert Monitor
 - for UNIX 198
 - for Windows 198
 - conflict detection
 - levels of 46
 - overview 46
 - peer-to-peer replication 7
 - planning 7
 - requirements 38
 - update-anywhere replication 7
 - conflicts
 - preventing 7
 - connecting
 - to iSeries server 19
 - connectivity
 - between DB2 operating systems 19
 - failure recovery for control tables 179
 - consistent-change-data (CCD) tables
 - adding UOW columns 68
 - external
 - multi-tier replication 71
 - internal
 - multiple targets 69
 - locks on 9
 - non-DB2 relational data sources
 - using CCD tables 35
 - nonrelational data sources
 - maintaining CCD tables 52
 - using CCD tables 33
 - replication sources 71
 - structure
 - Capture control server 390
 - target server 454
 - usage
 - history or audit 68
 - multi-tier replication 71
 - contact groups 189
 - contacting IBM 465
 - contacts
 - description 189
 - contacts and contact groups for monitoring, defining 200
 - contacts and contact groups, defining 200
 - control tables
 - Apply control server 412
 - authorization requirements for iSeries 28
 - Capture server 380
 - CCD (consistent-change-data)
 - Capture control server 390
 - target server 454
 - CD (change-data) 391
 - connectivity failure recovery 179
 - creating
 - for non-DB2 relational sources 22
 - in IASP groups 22
 - multiple database operating system 21
 - multiple database partitions 23
 - multiple sets 23
 - on iSeries 22, 327
 - on Linux, UNIX, Windows 21
 - dynamic 175
 - granting authority for iSeries 13, 333
 - control tables (*continued*)
 - I/O error recovery 179
 - IBMSNAP_APPENQ 413
 - IBMSNAP_APPLY_JOB 413
 - IBMSNAP_APPLYTRACE 417
 - IBMSNAP_APPLYTRAIL 418
 - IBMSNAP_APPPARMS 414
 - IBMSNAP_AUTHTKN 388
 - IBMSNAP_CAPENQ 382
 - IBMSNAP_CAPMON
 - pruning 180
 - structure 382
 - IBMSNAP_CAPPARMS
 - structure 384
 - IBMSNAP_CAPSCHEMAS 387
 - IBMSNAP_CAPTRACE
 - pruning 180
 - structure 389
 - IBMSNAP_PARTITIONINFO 392
 - IBMSNAP_PRUNCNTL 393
 - IBMSNAP_PRUNE_LOCK 395
 - IBMSNAP_PRUNE_SET 396
 - IBMSNAP_REG_EXT 396
 - IBMSNAP_REG_SYNCH 404
 - IBMSNAP_REGISTER 398
 - IBMSNAP_RESTART 405
 - IBMSNAP_SEQTABLE 407
 - IBMSNAP_SIGNAL 407
 - IBMSNAP_SUBS_COLS 423
 - IBMSNAP_SUBS_EVENT 424
 - IBMSNAP_SUBS_SET 429
 - IBMSNAP_SUBS_STMETS 434
 - IBMSNAP_UOW 410
 - maintaining 175
 - Monitor control server
 - IBMSNAP_ALERTS 437
 - IBMSNAP_CONDITIONS 438
 - IBMSNAP_CONTACTGRP 443
 - IBMSNAP_CONTACTS 444
 - IBMSNAP_GROUPS 445
 - IBMSNAP_MONENQ 445
 - IBMSNAP_MONPARMS 445
 - IBMSNAP_MONSERVERS 447
 - IBMSNAP_MONTRACE 449
 - IBMSNAP_MONTRAIL 449
 - IBMSNAP_SUSPENDS 451
 - IBMSNAP_TEMPLATES 452
 - list 436
 - pruning 178
 - quick reference
 - Apply control server 412
 - at a glance 373
 - Capture server 380
 - target server 452
 - rebinding, packages and plans 175
 - reorganizing 176
 - Replication Alert Monitor,
 - creating 199
 - revoking authority for iSeries 351
 - RUNSTATS utility 175
 - static 176
 - storage requirements 5
 - SUBS_MEMBR (subscription members) 425
 - target server 452
 - control_server parameter 116
 - copying replication configurations 167

- copyonce parameter 116
- COPYONCE parameter 357
- correlation ID 49
- CRTDPRTBL command 327
- CRTJRN command 29
- CRTJRNRCV command 29
- CTLSVR parameter 355
- current receiver size 3, 30
- customizing, SQL scripts 235

D

- data
 - advanced subsetting techniques 85
 - displaying historical 228
 - manipulating 89
 - preventing double-deletes 49
 - retrieving from source tables 180
 - subsetting
 - during registration 85
 - using predicates 87
 - using views 86
 - using views to specify predicates 87
 - transforming
 - at registration 89
 - at subscription 89
 - creating computed columns 91
 - renaming columns 76, 91
- data blocking 58
- data consistency 72
- data encryption restrictions 81
- data types
 - mapping between columns 76
 - replicating
 - large objects (LOB) 82
- data-sharing mode 136
- DB2 Extenders
 - restrictions 82
- DB2 replication
 - authorization requirements 13
- DB2 tables
 - registering 33
- DB2 UDB for z/OS
 - planning 10
- DB2 views
 - registering 51
- db2_subsystem parameter 117
- DB2CODEPAGE environment
 - variable 10, 24
- DB2DBDFT environment variable 24
- DB2INSTANCE environment variable 24
- DBADM 13
- DBCLOB (double-byte character large object)
 - replication considerations 82
- DBHEAP configuration parameter 24
- deactivating
 - registered objects 142
 - subscription sets 57, 158
- defaults
 - for Apply parameters (iSeries) 113
 - for Apply parameters (Linux, UNIX, Windows, z/OS) 113, 115
 - for Capture parameters (iSeries) 95
 - for Capture parameters (Linux, UNIX, Windows, z/OS) 95

- defaults (*continued*)
 - for Capture parameters (UNIX, Windows, z/OS) 97
- delay parameter 117
- DELAY parameter 356
- delete journal receiver exit program 173
- delete journal receiver exit routine
 - about 32
- delimiters, in generated SQL scripts 235
- diagnostic files
 - storage 6, 7
- difference table 183
- differential refresh replication 37
- disk space
 - requirements 3
- distinct data types 81
- distributed recovery points 163
- documentation
 - accessible 466
 - ordering 465
 - Web site 465
- double-byte character large object (DBCLOB)
 - replication considerations 82
- double-deletes 49
- DPR registrations (iSeries)
 - adding 287
 - removing 347
- DSPJRN command 233
- dynamic control tables 175

E

- e-mail notification, replication 195
- editing, SQL scripts 235
- email_server parameter, Replication Alert Monitor 208
- ENDDPRAPY command 328
- ENDDPRCAP command 108, 331
- environment variables
 - Capture program 24
 - DB2CODEPAGE 10, 24
 - DB2DBDFT 24
 - DB2INSTANCE 24
 - LIBPATH 24
- errors
 - monitor_errors parameter 208
- monitoring with alert conditions 189
- replication
 - alert conditions, APPLY_ERRORS 192
 - alert conditions, CAPTURE_ERRORS 192
 - alert conditions, QAPPLY_ERRORS 192
 - alert conditions, QCAPTURE_ERRORS 192
 - SQL 192
- errwait parameter 117
- event publishing
 - authorization requirements
 - Replication Alert Monitor 198
- event publishing commands
 - asnpwd 264
 - asnsct 268
 - asnsdrop 271
 - asnslist 271

- event publishing commands (*continued*)
 - asntdiff 272
 - asntrc 275
 - asntrep 282
- event-based scheduling 61
- events, coordinating 161
- existing tables as targets 74
- exit routines
 - ASNDONE
 - using 123, 124
 - ASNLOAD
 - customizing 129
 - for iSeries 131
 - for UNIX 126
 - for Windows 126
 - for z/OS 128
 - using 126
- delete journal receiver (iSeries) 32
- external CCD tables
 - multi-tier replication 71

F

- files
 - *.APP.log 118
 - *.CAP.log 100
 - *.err 121
 - *.sqs 121
 - asndone.smp 123
 - asnload.ini 130
- fragmentation
 - horizontal
 - at the source 37
 - at the target 75
 - peer-to-peer replication 7
 - update-anywhere replication 7
 - vertical
 - at the source 37
 - at the target 75
- FRCFRQ parameter 365
- full-refresh copying
 - Apply for iSeries 48, 355
 - registration option 37
- FULLREFPGM parameter 355

G

- gap detection 68
- generated SQL scripts 235
- global record 399
- GRTDPRAUT command
 - granting privileges to SQL packages 27
 - syntax 333
- GRTOBJAUT command 27

H

- heterogeneous replication
 - registering sources 35
 - restrictions
 - aggregate tables 68
 - CCD tables 38
 - multi-tier replication 71
 - update-anywhere 42, 72

- history data
 - CCD tables 68
 - source data 38
- horizontal (row) subsetting
 - at the source 37
 - at the target 75

I

- I/O error recovery, control tables 179
- IASP groups 22
- IBMSNAP_ALERTS control table 437
- IBMSNAP_APPENQ table 413
- IBMSNAP_APPLY_JOB table 413
- IBMSNAP_APPLYTRACE table
 - pruning 178
 - structure 417
- IBMSNAP_APPLYTRAIL table
 - pruning 178
 - structure 418
- IBMSNAP_APPPARMS table 414
 - using 122
- IBMSNAP_AUTHTKN table 388
- IBMSNAP_CAPENQ table 382
- IBMSNAP_CAPMON table
 - pruning 180
 - structure 382
- IBMSNAP_CAPPARMS table
 - structure 384
- IBMSNAP_CAPSCHEMAS table 387
- IBMSNAP_CAPTRACE table
 - pruning 180
 - structure 389
- IBMSNAP_CONDITIONS control table 438
- IBMSNAP_CONTACTGRP control table 443
- IBMSNAP_CONTACTS control table 444
- IBMSNAP_GROUPS control table 445
- IBMSNAP_MONENQ control table 445
- IBMSNAP_MONPARMS control table 445
- IBMSNAP_MONSERVERS control table 447
- IBMSNAP_MONTRACE control table 449
- IBMSNAP_MONTRAIL control table 449
- IBMSNAP_PARTITIONINFO table 392
- IBMSNAP_PARTITIONINFOtable 392
- IBMSNAP_PRUNCNTL table 393
- IBMSNAP_PRUNE_LOCK table 395
- IBMSNAP_PRUNE_SET table 396
- IBMSNAP_REG_EXT table 396
- IBMSNAP_REG_SYNCH table 404
- IBMSNAP_REGISTER table 398
- IBMSNAP_RESTART table 405
- IBMSNAP_SEQTABLE table 407
- IBMSNAP_SIGNAL table
 - structure 407
- IBMSNAP_SUBS_COLS table 423
- IBMSNAP_SUBS_EVENT table
 - structure 424
- IBMSNAP_SUBS_SET table 429
- IBMSNAP_SUBS_STMTS table 434
- IBMSNAP_SUSPENDS control table 451

- IBMSNAP_TEMPLATES control table 452
- IBMSNAP_UOW table
 - pruning 410
 - structure 410
- IMS data sources
 - maintaining CCD tables 52
 - registering 33
 - using CCD tables 33
- IMS DataPropagator 33
- inactive subscription sets 57
- INACTMSG parameter 356
- inamsg parameter 118
- Independent Auxiliary Storage Pool (IASP) groups 22
- indexes
 - target tables 77
- inner-joins as sources 49
- internal CCD tables
 - multiple targets 69
- interval timing 61
- invocation parameters
 - Analyzer
 - for iSeries 320
 - Apply program
 - for iSeries 112, 354
 - for UNIX 115
 - for Windows 115
 - for z/OS 115
 - Capture program
 - for iSeries 93, 94, 323, 361
 - for UNIX 97
 - for Windows 97
 - for z/OS 97
 - Replication Alert Monitor
 - for UNIX 257
 - for Windows 257
 - for z/OS 257
 - replication commands
 - for iSeries 288, 297, 311, 328, 329, 331, 333, 341, 343, 347, 348, 350, 352, 354, 361, 368
- INZDPRCAP command 341
- iSeries data sources
 - with remote journaling 47
- iSeries server
 - connecting to 19

J

- JCL
 - starting the Apply program 133
 - starting the Capture program 133
 - starting the Replication Alert Monitor 133
- JCL batch mode 133
- JOBID parameter 354, 361
- JOIN_UOW_CD column 87
- joins as sources 49
- journal jobs
 - checking status 233
- journal message queues 31
- journal receivers
 - access 173
 - creating for source tables 29
 - current, size 3
 - delete journal receiver exit routine 32

- journal receivers (*continued*)
 - maintaining 172
 - managing 30
 - system management 30
 - threshold 30
 - user management 31
- journal signal tables
 - CAPSTOP 166
 - stopping 163
- journals
 - creating 29
 - creating for source tables 29
 - default message queue 31
 - entry types 463
 - managing 30
 - QSQRJRN journal 28
 - registering as sources 33
 - setup 28
 - starting 29
 - using 28
 - using remote journal function 47
- JRN parameter 362

L

- LAG parameter 364
- lag_limit parameter 100
- LANG variable
 - setting 10
- large object (LOB)
 - replication considerations 82
- large replication jobs 58
- latency
 - Apply program 232
 - Capture program 230
- legal notices 467
- LIBPATH 24
- load from cursor function 129
- loadxit parameter 118
- LOB (large object)
 - replication considerations 82
 - update-anywhere restrictions 72
- locks
 - on CCD tables 9
- log
 - planning impact to 8
- log records
 - archived before captured 3
 - compression dictionaries (z/OS) 174
 - maintaining 172
 - multi database partitions 172
 - retaining 172
- LOGBUFSZ configuration parameter 24
- LOGFILSIZ configuration parameter 24
- logging requirements
 - DB2 source servers 3
 - non-DB2 relational source servers 8
 - target servers 4
- logical partitioning keys
 - description 42
- LOGPRIMARY configuration parameter 24
- logreuse parameter (for Apply) 118
- logreuse parameter (for Capture) 100
- LOGSECOND configuration parameter 24
- logstdout parameter (for Apply) 119

- logstdout parameter (for Capture) 100
- long name support
 - planning 10
- LONG VARCHAR data types 81
- LONG VARGRAPHIC data types 81

M

- manipulating data
 - at registration 89
 - at subscription 89
 - creating computed columns 91
 - renaming columns 76, 91
- mapping
 - data types between tables 76
 - source columns to target columns 76
 - sources to targets 63
- master tables (update-anywhere)
 - overview 72
 - recapturing changes 42
- max_notification_minutes parameter, Replication Alert Monitor 208
- max_notifications_per_alert parameter, Replication Alert Monitor 208
- MAX_SYNCH_MINUTES, data blocking 58
- MAXAPPLS configuration parameter 24
- MEMLMT parameter 363
- memory
 - alert conditions
 - APPLY_MEMORY 192
 - CAPTURE_MEMORY 192
 - QAPPLY_MEMORY 192
 - QCAPTURE_MEMORY 192
 - Apply program 3
 - batch jobs 1
 - Capture program 1
 - planning 1
 - reading log records 1
 - registrations 1
 - Replication Alert Monitor 197
 - subscription sets 3
 - transactions 1
 - using IBMSNAP_CAPMON table to tune 1
- memory_limit parameter
 - overview 101
 - tuning 1
- merging
 - subscription sets 154
 - triggers 8
- message queues, for journals 31
- messages 207, 230, 231
- Microsoft SQL Server
 - replication restrictions 38
- migration
 - planning 1
- mini-cycles 58
- MODIFY command 133
- monitor control server
 - IBMSNAP_SUSPENDS control table 451
 - IBMSNAP_TEMPLATES control table 452
- Monitor control server
 - IBMSNAP_ALERTS control table 437

- Monitor control server (*continued*)
 - IBMSNAP_CONDITIONS control table 438
 - IBMSNAP_CONTACTGRP control table 443
 - IBMSNAP_CONTACTS control table 444, 445
 - IBMSNAP_MONENQ control table 445
 - IBMSNAP_MONPARMS control table 445
 - IBMSNAP_MONSERVICES control table 447
 - IBMSNAP_MONTRACE control table 449
 - IBMSNAP_MONTRAIL control table 449
 - list of control tables 436
- Monitor program
 - messages 207
 - printing 207
- monitor qualifier
 - replication 189
- Monitor qualifiers, naming rules 237
- monitor_errors parameter, Replication Alert Monitor 208
- monitor_interval parameter (for Capture) 101
- monitor_limit parameter 101
 - Replication Alert Monitor 208
- monitor_path parameter, Replication Alert Monitor 208
- monitoring
 - for iSeries 233
 - historical trends 228
 - replication 189
 - status of programs 233
- MONITV parameter 363
- MONLMT parameter 363
- multi database partition
 - log records 172
- multi-tier replication
 - defining subscription sets 71
- multiple database partitions
 - Capture 28
- multiple target tables 69
- MVS console 133, 134

N

- names
 - Apply qualifier rules 237
 - Capture schema rules 237
 - for Windows services 238
 - Monitor qualifier rules 237
 - of Capture triggers 8
 - subscription sets 150
- national language support (NLS) 10
- network connectivity 19
- nicknames
 - for load from cursor function 129
 - registering 35
 - restrictions
 - aggregate tables 68
 - multi-tier replication 71
 - update-anywhere 42, 72
 - with CCD tables 38

- NLS (national language support) 10
- non-DB2 relational data sources
 - locks 9
 - registering 35
 - restrictions
 - aggregate tables 68
 - multi-tier replication 71
 - update-anywhere 42, 46, 72
 - source servers 8
 - using CCD tables 35
- non-DB2 relational servers
 - connecting 20
- nonrelational data sources
 - maintaining CCD tables 52
 - using CCD tables 33
- notify parameter 119

O

- objects
 - changing attributes 140
 - deactivating 142
 - reactivating 143
 - registering 139
 - stop capturing changes 142
- On-demand reporting 227
- operating
 - Replication Alert Monitor 261
- opt4one parameter 119
- OPTSNGSET parameter 357
- overriding attributes (iSeries)
 - Capture program 342
- OVRDPRCAPA command 342

P

- packages, rebinding 175
- parameters
 - Replication Alert Monitor
 - alert_prune_limit 208
 - autoprun 208
 - default values 208
 - description 208
 - email_server 208
 - max_notification_minutes 208
 - max_notifications_per_alert 208
 - monitor_errors 208
 - monitor_limit 208
 - monitor_path 208
 - runonce 208
 - trace_limit 208
- parameters, invocation
 - Analyzer
 - for iSeries 320
 - Apply program
 - for iSeries 112, 354
 - for UNIX 115
 - for Windows 115
 - for z/OS 115
 - Capture program
 - for iSeries 323, 361
 - for UNIX 97
 - for Windows 97
 - for z/OS 97
 - Replication Alert Monitor
 - for UNIX 257

- parameters, invocation (*continued*)
 - Replication Alert Monitor (*continued*)
 - for Windows 257
 - for z/OS 257
 - replication commands
 - for iSeries 288, 297, 311, 328, 329, 331, 333, 341, 343, 347, 348, 350, 352, 354, 361, 368
- password file
 - creating 264, 275
- password files
 - storing 17
- peer-to-peer replication
 - conflict detection 7
- performance
 - tuning 11
- planning
 - coexistence of triggers 8
 - conflict detection 7, 46
 - locks on CCD tables 9
 - log impact 8
 - memory 1
 - migration 1
 - storage requirements 3
 - transaction throughput rates 8
- plans, rebinding 175
- point-in-time tables
 - structure 456
 - usage 67
- predicates
 - defining for target tables 75
 - subsetting 87
- PREDICATES column 87
- prefix, before-image 41
- primary keys
 - logical partitioning 42
 - relative record numbers for iSeries 48
 - used as target key 77
- printing
 - Apply program
 - messages 231
 - Capture program
 - printing 230
 - Monitor program
 - messages 207
- promoting
 - replication configurations 167
- prune_interval parameter 101
- pruning
 - CD (change-data) tables 177
 - control tables 178
 - IBMSNAP_APPLYTRACE table 178
 - IBMSNAP_APPLYTRAIL table 178
 - IBMSNAP_CAPMON table 180
 - IBMSNAP_CAPTRACE table 180
 - IBMSNAP_UOW table 410
 - signal (SIGNAL) table 181
 - UOW (unit-of-work) table 177
- pwdfile parameter 119

Q

- Q Apply program
 - alert conditions 192
- Q Capture program
 - alert conditions 192

- Q replication
 - authorization requirements
 - for Replication Alert Monitor 198
 - control tables
 - list, Replication Alert Monitor 436
- Q replication commands
 - ansrcrt 268
 - asnsdrop 271
 - asnslst 271
 - asnspwd 264
 - asnstrc 275
 - asntdiff 272
 - asntrep 282

R

- RCVJRNE command 30
- reactivating
 - objects 143
 - registrations 143
 - tables 143
- read dependencies 46
- readers' comment form 466
- rebinding, packages and plans 175
- recapturing changes (update-anywhere) 42
- receiver size, current 3
- recovery points, distributed 163
- referential integrity 72
- registering
 - DB2 tables 33
 - IMS data sources 33
 - non-DB2 relational data sources 35
 - objects 139
 - options for sources
 - after-image columns 38
 - before-image columns 38
 - before-image prefix 41
 - change-capture replication 37
 - column (vertical) subsetting 37
 - conflict detection 46
 - full-refresh copying 37
 - recapturing changes (update-anywhere) 42
 - relative record numbers 48
 - row (horizontal) subsetting 37
 - stop Capture on error 41
 - updates as deletes and inserts 42
 - using remote journals 47
 - tables 139
 - views
 - overview 49, 51
 - procedure 139
- registrations
 - adding 287
 - adding columns 140
 - attributes, changing 140
 - deactivating 142
 - reactivating 143
 - removing 144, 347
 - stop capturing changes 142
- registry variables
 - DB2CODEPAGE 10, 24
 - DB2DBDFT 24
 - DB2INSTANCE 24
- reinitializing 205

- reinitializing Capture program
 - for UNIX 108
 - for Windows 108
 - for z/OS 108
- relative record numbers
 - as primary key for iSeries 48
 - support for iSeries 48
 - used as target key 77
- relative timing 61
- remote journals as sources 47
- remote source tables 47
- renaming columns 76, 91
- reorganizing
 - control tables 176
- replica tables
 - defining read-write targets 72
 - definition 65
 - recapturing changes 42
 - structure 457
- Replication Alert Monitor 200, 205, 206
 - alert conditions
 - e-mail notifications 195
 - events 189
 - list 192
 - overview 192
 - sending alerts to z/OS
 - console 196
 - status 189
 - thresholds 189
 - alerts 189
 - authorization requirements 198
 - changing alert conditions 202
 - changing runtime parameters 211
 - communicating with
 - Apply program 225
 - Capture 225
 - Replication Center 224
 - contact groups 189
 - contacts 189
 - control tables
 - IBMSNAP_ALERTS 437
 - IBMSNAP_CONDITIONS 438
 - IBMSNAP_CONTACTGRP 443
 - IBMSNAP_CONTACTS 444
 - IBMSNAP_GROUPS 445
 - IBMSNAP_MONENQ 445
 - IBMSNAP_MONPARMS 445
 - IBMSNAP_MONSERVICES 447
 - IBMSNAP_MONTRACE 449
 - IBMSNAP_MONTRAIL 449
 - creating control tables 199
 - description 189
 - for UNIX
 - binding 198
 - checking status 227
 - operating 261
 - starting 461
 - for Windows
 - binding 198
 - checking status 227
 - operating 261
 - starting 461
 - for z/OS
 - checking status 227
 - operating 261
 - memory 197
 - monitoring replication, overview 189

- Replication Alert Monitor (*continued*)
 - operational errors 212
 - parameters
 - alert_prune_limit 208
 - autoprunce 208
 - default values 208
 - descriptions 208
 - email_server 208
 - max_notification_minutes 208
 - max_notifications_per_alert 208
 - monitor_errors 208
 - monitor_interval 208
 - monitor_limit 208
 - monitor_path 208
 - runonce 208
 - trace_limit 208
 - pruning control tables 212
 - scheduling 219, 220
 - selecting alert conditions 201
 - setting up 197
 - specifying notification criteria 212
 - specifying run frequency 211
 - stopping 207
- Replication Analyzer
 - for iSeries
 - creating SQL packages 27
 - invocation parameters 320
- Replication Center
 - communicating with
 - Apply program 221
 - Capture program 221
 - Capture triggers 221
 - Replication Alert Monitor 224
 - connectivity 19
 - promote functions 167
- replication commands
 - \$TA JES2
 - Apply for z/OS 220
 - Capture for z/OS 220
 - ADDJOBSCDE 220
 - asnslist 271
 - asntdiff 272
 - asntrep 282
 - AT 219, 220
 - AT NetView
 - Apply for z/OS 220
 - Capture for z/OS 220
 - backup database 24
 - CRTJRNRCV 29
 - DSPJRN 233
 - for iSeries
 - ADDDPRREG 287
 - ADDDPRSUB 295
 - ADDDPRSUBM 310
 - ANZDPR 319
 - ANZDPRJRN 32
 - CHGDPRCAPA 322
 - CHGJRN 31
 - CRTPRTBL 327
 - CRTJRN 29
 - ENDDPRAPY 328
 - ENDDPRCAP 108, 331
 - GRTDPRAUT 27, 333
 - GRTOBJAUT 27
 - INZDPRCAP 341
 - OVRDPRCAPA 342
 - RCVJRNE 30

- replication commands (*continued*)
 - for iSeries (*continued*)
 - RMVDPRREG 347
 - RMVDPRSUB 348
 - RMVDPRSUBM 350
 - RVKDPRAUT 351
 - SBMJOB 220
 - STRDPRAPY 113, 353
 - STRDPRCAP 359
 - STRJRNPf 29
 - WRKDPRTRC 366
 - WRKJOB 233
 - WRKSBMJOB 233
 - WRKSBSJOB 233
 - for UNIX
 - asnanalyze 254
 - asnmcmd 261
 - for Windows
 - asnanalyze 254
 - asnmcmd 261
 - for z/OS
 - asnmcmd 261
 - MODIFY 133
 - update database configuration 24
- replication environments
 - copying 167
- replication events coordination 161
- replication services
 - creating 216
 - description 215
 - display name 215
 - dropping 218
 - listing 271
 - name 215
 - starting 217
 - stopping 217
 - viewing 217
- replication sources
 - CCD (consistent-change-data)
 - tables 71
 - joins 49
 - maintaining CCD tables 52
 - mapping to targets 63
 - registering
 - columns 37
 - DB2 tables 33
 - IMS data sources 33
 - non-DB2 relational data
 - sources 35
 - rows 37
 - views 51
 - subscribing to 55
 - RESTART parameter 361
 - restrictions
 - abstract data types 81
 - ASCII tables 459
 - CCD tables 72
 - column names, limits 38
 - data encryption 81
 - DB2 Extenders large objects 82
 - distinct data types 81
 - existing target tables 74
 - heterogeneous replication 38, 71, 72
 - LOB data types 72
 - LONG columns in Oracle tables 81
 - LONG VARCHAR data types 81
 - LONG VARGRAPHIC data types 81

- restrictions (*continued*)
 - Microsoft SQL Server 38
 - non-DB2 relational data sources 42, 46
 - Oracle sources 81
 - spatial data types 81
 - stored procedures 90
 - Sybase 38
 - Unicode tables 459
 - user-defined data types 81
 - views 51
 - WHERE clause 75
- resuming 206
 - Capture program
 - for UNIX 109
 - for Windows 109
 - for z/OS 109
- RETAIN parameter 364
- retention_limit parameter 102
- RMVDPRREG command 347
- RMVDPRSUB command 348
- RMVDPRSUBM command 350
- roll-forward recovery 24
- row (horizontal) subsetting
 - at the source 37
 - at the target 75
- row-capture rules 37
- ROWID 82
- rows
 - available for replication 37
 - defining in target table 75
 - registering in source table 37
 - subsetting
 - at the source 37
 - at the target 75
- RRN 48
- RTYWAIT parameter 356
- run-time processing 61, 90
- running, SQL scripts 235
- runonce parameter, Replication Alert Monitor 208
- RUNSTATS utility 175
- RVKDPRAUT command 351

S

- SBMJOB command 220
- scheduling
 - replication programs 219
 - subscription sets 61
- schemas
 - changing 145
 - naming rules 237
- SCM (Service Control Manager)
 - creating replication services 216
 - description 215
 - dropping replication services 218
 - starting replication services 217
 - stopping replication services 217
 - viewing replication services 217
- screen readers 466
- Service Control Manager (SCM)
 - creating replication services 216
 - description 215
 - dropping replication services 218
 - starting replication services 217
 - stopping a replication service 217

- Service Control Manager (SCM)
 - (continued)
 - viewing replication services 217
- setting environment variables
 - Capture program 24
- setting up
 - Apply programs
 - for UNIX 24
 - for Windows 24
 - Capture programs
 - for UNIX 24
 - for Windows 24
 - journals 28
 - Replication Alert Monitor 198
- signal (SIGNAL) table
 - pruning 181
- SIGNAL (signal) table
 - pruning 181
- signals
 - CAPSTART 165
 - CAPSTOP 166
 - setting distributed recovery points 163
 - STOP 162, 163
 - USER 161
- sleep parameter 120
- sleep_interval parameter 103
- source logs, maintaining 172
- source servers
 - DB2
 - log impact 3
 - non-DB2 relational
 - log impact 8
- source systems, maintaining 171
- source tables
 - adding columns 140
 - creating journals for 29
 - maintaining 171
 - retrieving lost data 180
- sources
 - CCD (consistent-change-data) tables 71
 - maintaining CCD tables 52
 - mapping to targets 63
 - registering
 - DB2 tables 33
 - IMS data sources 33
 - non-DB2 relational 35
 - views 49, 51
 - registering columns 37
 - registering rows 37
 - registration options
 - after-image columns 38
 - before-image columns 38
 - before-image prefix 41
 - change-capture replication 37
 - column (vertical) subsetting 37
 - conflict detection 46
 - full-refresh copying 37
 - recapturing changes (update-anywhere) 42
 - relative record numbers 48
 - row (horizontal) subsetting 37
 - stop Capture on error 41
 - updates as deletes and inserts 42
 - using remote journals 47
 - subscribing to 55
- spatial data types 81
- special data types
 - replicating
 - large objects (LOB) 82
- spill files
 - storage for Apply 6
 - storage for Capture 6
- spillfile parameter 120
- splitting
 - subscription sets 151
- SQL files, editing 235
- SQL packages
 - creating for Apply program 27
 - creating for Capture program 26
 - creating for Replication Analyzer 27
- SQL replication 227
 - planning overview 1
- SQL replication commands
 - asnpwd 264
 - ansrcr 268
 - asnsdrop 271
 - asnrtr 275
- SQL replication
 - componentcommunication 221
- SQL scripts 235
- SQL statements
 - defining for subscription set 61
 - run-time processing 90
- sqlerrcontinue parameter 121
- staged replication 71
- staging data 71
- starting 205
 - Apply program
 - for iSeries 112, 353
 - for UNIX 111, 461
 - for Windows 111, 461
 - for z/OS 111, 134
 - Capture program
 - for iSeries 94, 359
 - for UNIX 93, 461
 - for Windows 93, 461
 - for z/OS 93
 - Replication Alert Monitor
 - for UNIX 461
 - for Windows 461
- startmode parameter 103
- static control tables 176
- status
 - Apply program 227, 233
 - Capture program 227, 233
 - journal jobs 233
 - Replication Alert Monitor 227
- stop Capture on error option 41
- stop capturing changes 142
- STOP signals 162, 163
- stopping
 - Apply program
 - for iSeries 123, 328
 - for UNIX 123
 - for Windows 123
 - for z/OS 123
 - Capture program
 - for iSeries 108, 331
 - for UNIX 108
 - for Windows 108
 - for z/OS 108
- stopping (continued)
 - Replication Alert Monitor
 - for UNIX 261
 - for Windows 261
 - for z/OS 261
- storage
 - Apply diagnostic files 6
 - Apply spill files 6
 - Capture diagnostic files 6
 - Capture spill files 6
 - CD table 5
 - control tables 5
 - database log and journal data 3
 - diagnostic files 7
 - requirements 3
 - target tables 5
 - UOW table 5
- stored procedures
 - defining for subscription set 61
 - manipulating data 90
- STRDPRAPY command 113, 353
- STRDPRCAP command 359
- STRJRNPF command 29
- SUBNFYPGM parameter 356
- SUBS_EVENT (subscription events) table
 - posting events 61
- SUBS_MEMBR (subscription members) table 129, 425
- subscribing to sources 55
- subscription cycle 58
- subscription events (SUBS_EVENT) table
 - posting events 61
- subscription members (SUBS_MEMBR) table 129, 425
- subscription sets
 - activation level 57
 - adding 295
 - adding members 63, 147
 - changing
 - Apply qualifiers 157
 - attributes 149
 - names 150
 - columns 75
 - creating 55
 - creating new 146
 - data consistency 72
 - deactivating 158
 - disabling members 148
 - enabling members 148
 - merging 154
 - mini-cycles 58
 - multi-tier replication 71
 - number of Apply qualifiers 54
 - processing mode 60
 - referential integrity 72
 - removing 160, 348
 - rows 75
 - run-time processing statements 90
- scheduling
 - event-based 61
 - time-based 61
- splitting 151
- SQL statements 61
- stored procedures 61
- update-anywhere replication 72
- subscription-set members
 - adding 63, 147, 310

- subscription-set members *(continued)*
 - applying subset of columns 75
 - applying subset of rows 75
 - defining target key 77
 - disabling 148
 - enabling 148
 - mapping between columns 76
 - mapping data types 76
 - multi-tier replication 71
 - number per subscription set 54
 - removing 350
 - selecting target types 65
 - update-anywhere replication 72
- subsetting
 - advanced techniques
 - during registration 85
 - using predicates 87
 - columns at target 75
 - registered columns 37
 - registered rows of changes 37
 - rows of changes at target 75
 - source data
 - using views 86
- suspending 206
 - Capture program
 - for UNIX 109
 - for Windows 109
 - for z/OS 109
- Sybase
 - replication restrictions 38
- synchronization
 - asntdiff and asntrepair utilities 183
- SYSADM 13
- system change journal management 30
- system commands
 - asnacmd 253
 - asnapply 247
 - asncap 239
 - asnccmd 244
 - asnpwd 264
 - ansrcrt 268
 - ansndrop 271
 - asnslist 271
 - asntdiff 272
 - asntrc 275
 - asntrep 282
- system-started tasks 133

T

- table differencing utility 183, 272
- table repair utility 186, 282
- table-mode processing 4, 60
- tables
 - adding columns 140
 - AUTHTKN (Apply-qualifier cross-reference) 388
 - base aggregate 453
 - CCD (consistent-change-data)
 - Capture control server 390
 - target server 454
 - CD (change-data) 391
 - change aggregate 453
 - changing attributes 140
 - conflict detection for 7
 - control tables
 - connectivity failure recovery 179

- tables *(continued)*
 - control tables *(continued)*
 - dynamic 175
 - I/O error recovery 179
 - maintaining 175
 - pruning 178
 - reorganizing 176
 - RUNSTATS utility 175
 - static 176
 - deactivating 142
 - IBMSNAP_ALERTS 437
 - IBMSNAP_APPENQ 413
 - IBMSNAP_APPLY_JOB 413
 - IBMSNAP_APPLYTRACE 417
 - IBMSNAP_APPLYTRAIL 418
 - IBMSNAP_APPPARMS 414
 - IBMSNAP_CAPENQ 382
 - IBMSNAP_CAPMON 180, 382
 - IBMSNAP_CAPPARMS 384
 - IBMSNAP_CAPSCHEMAS 387
 - IBMSNAP_CAPTRACE 180, 389
 - IBMSNAP_CONDITIONS 438
 - IBMSNAP_CONTACTGRP 443
 - IBMSNAP_CONTACTS 444
 - IBMSNAP_GROUPS 445
 - IBMSNAP_MONENQ 445
 - IBMSNAP_MONPARMS 445
 - IBMSNAP_MONSERVERS 447
 - IBMSNAP_MONTRACE 449
 - IBMSNAP_MONTRAIL 449
 - IBMSNAP_PARTITIONINFO 392
 - IBMSNAP_PRUNCNTL 393
 - IBMSNAP_PRUNE_LOCK 395
 - IBMSNAP_PRUNE_SET 396
 - IBMSNAP_REG_EXT 396
 - IBMSNAP_REG_SYNCH 404
 - IBMSNAP_REGISTER 398
 - IBMSNAP_RESTART 405
 - IBMSNAP_SEQTABLE 407
 - IBMSNAP_SIGNAL 407
 - IBMSNAP_SUBS_COLS 423
 - IBMSNAP_SUBS_EVENT 424
 - IBMSNAP_SUBS_SET 429
 - IBMSNAP_SUBS_STMTS 434
 - IBMSNAP_SUSPENDS 451
 - IBMSNAP_TEMPLATES 452
 - IBMSNAP_UOW 410
 - maintaining CCD tables 52
 - point-in-time 456
 - reactivating 143
 - registering
 - DB2 33
 - non-DB2 relational 35
 - procedure 139
 - removing registrations 144
 - replica 7, 457
 - stop capturing changes 142
 - SUBS_MEMBR (subscription members) 129, 425
 - target tables 181
 - maintaining 181
 - user copy 457
 - target indexes 77
 - target keys 77
 - target servers
 - log impact 4

- target tables
 - applying subset of columns 75
 - applying subset of rows 75
 - base aggregate
 - definition 65
 - structure 453
 - usage 68
 - CCD (consistent-change-data)
 - overview 65
 - structure 454
 - change aggregate
 - definition 65
 - structure 453
 - usage 68
 - defining columns 75
 - defining rows 75
 - defining target key 77
 - fragmenting 75
 - maintaining 181
 - mapping to sources 63
 - new columns for 91
 - point-in-time
 - definition 65
 - structure 456
 - usage 67
 - repairing 186
 - replica
 - conflict detection for 7
 - definition 65
 - structure 457
 - usage 72
 - storage requirements 5
 - table structures, quick reference 452
 - user copy
 - definition 65
 - structure 457
 - usage 67
 - user defined 65, 74
 - target-key columns
 - updating 78
 - term parameter (for Apply) 121
 - term parameter (for Capture) 104
 - termination characters, in generated SQL scripts 235
 - three-tier replication configuration 71
 - throughput
 - Apply program 232
 - Capture program 230
 - throughput rates
 - Capture triggers 8
 - time-based scheduling 61
 - tips
 - checking if Apply processed a set successfully 121
 - deleting rows from the Apply trail table 122
 - estimating use of space 3
 - using sleep versus copyonce parameters 116, 120
 - using stored procedures for additional processing of sets 123
 - using stored procedures with ASNDONE 124
 - verifying change capture began 93
 - trace facility
 - for iSeries 366
 - TRACE parameter 355

- trace_limit parameter
 - overview 104
 - Replication Alert Monitor 208
 - use with asnmnon command 257
- trademarks 469
- transaction throughput rates
 - Capture triggers 8
- transaction-mode processing 4, 60
- transactions
 - memory used by 1
- transforming data
 - at registration 89
 - at subscription 89
 - creating computed columns 91
 - renaming columns 76, 91
- translating data 10
- TRCLMT parameter 363
- triggers
 - capturing data 8
 - merging 8
- trlreuse parameter 122
- TRLREUSE parameter 357
- troubleshooting commands
 - WRKDPRTTRC 366
- TSO 133, 134
- tuning
 - commit_interval parameter 1
 - memory_limit parameter 1
 - performance 11

U

- Unicode tables 459
- unit-of-work (UOW) table
 - columns in CCD tables 68
 - pruning 177
 - storage requirements 5
- UOW (unit-of-work) table
 - columns in CCD tables 68
 - pruning 177
 - storage requirements 5
- UOW_CD_PREDICATES column 87
- update database configuration
 - command 24
- update-anywhere replication
 - conflict detection
 - overview 46
 - planning for 7
 - requirements 38, 46
 - defining subscription sets 72
 - fragmentation for 7
 - recapturing changes 42
- updated primary key columns 42
- updates
 - as deletes and inserts 42
 - conflicts 46
- user copy table
 - definition 65
 - structure 457
 - usage 67
- user IDs
 - authorization 14
 - for Apply program 15
 - for Capture program 14
 - for Capture triggers 16
 - password files 17
- USER parameter 354

- USER signals 161
- user-defined data types 81
- user-defined tables 65, 74
- utilities
 - table differencing 272
 - table repair 282

V

- vertical (column) subsetting
 - at the source 37
 - at the target 75
- views
 - changing attributes 140
 - registering
 - as sources 51
 - overview 49
 - procedure 139
 - restrictions 49, 51
 - using correlation ID 49

W

- WAIT parameter 361
- warm start, Capture program
 - for iSeries 361
 - for UNIX 103
 - for Windows 103
 - for z/OS 103
- warmns startmode 103
- warmsi startmode 103
- WHERE clause
 - PREDICATES column restriction 87
 - row subsets 75
- Windows service
 - creating 268, 271
- Windows Service Control Manager (SCM)
 - asnslst command 271
 - description 215
 - listing replication services 271
- Windows services names 238
- WRKDPRTTRC command 366
- WRKJOB command 233
- WRKSBJOB command 233
- WRKSBSJOB command 233

Z

- z/OS console
 - sending monitor alerts 196



Printed in USA

SC19-1030-00

